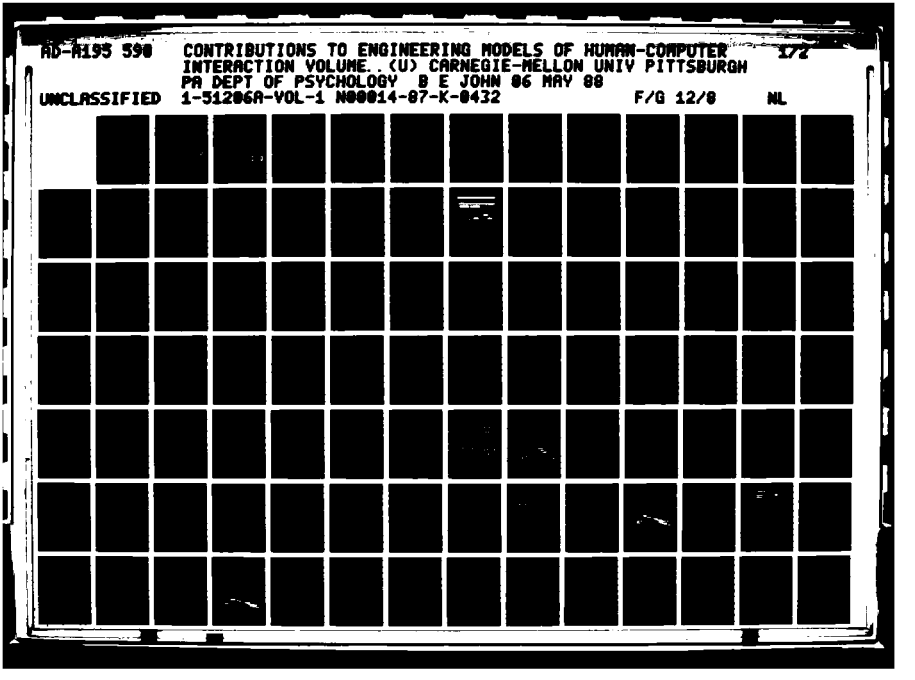
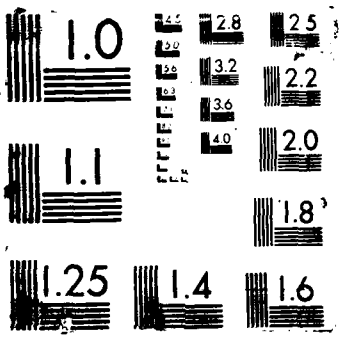

AD-A195 590 CONTRIBUTIONS TO ENGINEERING MODELS OF HUMAN-COMPUTER INTERACTION VOLUME..(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF PSYCHOLOGY B E JOHN 06 MAY 88
1/2
UNCLASSIFIED 1-51206A-VOL-1 N00014-87-K-0432 F/G 12/8 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

Contributions to Engineering Models of Human-Computer Interaction

Volume I

DEPARTMENT
of
PSYCHOLOGY

Carnegie Mellon University

(1)

Contributions to Engineering Models of Human-Computer Interaction

Volume I

Bonnie E. John

May 6, 1988

Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Report Number 1-51206A

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Approved for public release; distribution unlimited.

This research was, in part, supported by the Office of Naval Research, Perceptual Science Programs, Contract Number N00014-87-K-0432.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research or the US Government.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS N.A.
2a SECURITY CLASSIFICATION AUTHORITY Unclassified	3 DISTRIBUTION AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE N. A.	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) 1-51206A	5 MONITORING ORGANIZATION REPORT NUMBER(S) Same
6a NAME OF PERFORMING ORGANIZATION Department of Psychology Carnegie-Mellon University 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Office of Naval Research
6c ADDRESS (City, State, and ZIP Code) Pittsburgh, PA 15213-3890	7b ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION Office of Naval Research 8b OFFICE SYMBOL (If applicable) Code 1142PS	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N0001487-K-0432
8c ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 61153N 42 / PROJECT NO: RR 04209 / TASK NO: 0420901 / WORK UNIT ACCESSION NO: 4429005
11 TITLE (Include Security Classification) Contributions to Engineering Models of Human-Computer Interaction, Vol. I (unclassified)	
12 PERSONAL AUTHOR(S) John, Bonnie Elizabeth	
13a TYPE OF REPORT Technical 13b TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Year, Month, Day) 6 May, 1988 15 PAGE COUNT 63
16 SUPPLEMENTARY NOTATION	

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Human-computer interaction, cognitive modelling, GOMS

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This dissertation presents two engineering models of behavior at the human-computer interface: a model of immediate behavior and stimulus-response compatibility and a model of transcription typing. Formulated within the architecture of the Model Human Processor of Card, Moran and Newell, these models are able to make zero-parameter, quantitative predictions of human response time in their respective domains. They are also completely integrated, making good predictions about performance on a dual reaction-time/typing task. Parameters of the models are set using response time data from an abbreviation recall experiment. These parameters are then used to make predictions about response time in another abbreviation recall experiment, three classic stimulus-response experiments, and over 29 experiments that reflect robust phenomena associated with transcription typing. These models are the first to make succesful predictions across domain boundaries, both within tasks exhibiting stimulus-response compatibility and outside that paradigm to transcription typing.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL John J. O'Hare	22b TELEPHONE (Include Area Code) (202) 696-4502 22c OFFICE SYMBOL Code 1142PS

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

To my mother, Elizabeth C. John
and
to the memory of my father, Frederick John.
They laid the foundation of my work.

Acknowledgements

Many people have contributed to the completion of this thesis. First and foremost are my husband, Gary Pelton, and my children, Laura and Liza. Gary endured the tears, shared the triumphs, and gladly took care of Laura while I worked nights and weekends. Laura made me forget the pressures of grad school and let me escape to the worlds of Dr. Seuss and Sesame Street with her. Liza, though she came on the scene late in the process, mercifully let me sleep!

Allen Newell, my advisor, has shown me what it means to be a scientist. Words are inadequate to describe his example of honesty, critical thinking, hard work, respect, enthusiasm, and joy in discovery.

The faculty of the Psychology Department - especially my committee, John Anderson and Lynne Reder - provided an environment of challenge and support unsurpassed in the field.

The students and staff of the Psychology Department kept me moving, kept me laughing, and kept me on track, with my eyes on the light at the end of the tunnel.

John O'Hare at ONR provided monetary encouragement - just the type of encouragement a fledgling researcher needs.

Last, but not least, Linda McClory took care of my children with such love that their welfare was never in doubt. She freed my heart from worry and my mind from the details of everyday life.

Thank you all.

Preface

This dissertation is divided into two volumes. Volume I contains the body of the work and the bibliography. Most readers will require only Volume I. Volume II contains the appendices, over 150 algorithms used to predict reaction times in immediate behavior tasks. Volume II was written for archival purposes, intended only for those readers wishing to replicate my results. Volume II is on file in the Carnegie-Mellon library and is available upon request from the author.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Human-Computer Interaction

Human-computer interaction (HCI) promises to be an important area of application for cognitive science. It also provides a rich, easily manipulated, environment in which to study the development and performance of complex cognitive skills. Our basic interest is in the application side of this dual situation, bringing the advances in the psychologist's understanding of what makes systems easy to use and learn to the design of new and better computer systems. This thesis will be written from the application point of view, leaving to the end a discussion of its relevance for basic cognitive science.

Many aspects of how humans interact with computers are of interest: programming, communication between people via a computer link, the social impact of computers, decision-making with access to large databases--the list is endless. All of these areas, and indeed all of HCI, have the human-computer interface in common. It is the interface that allows the person to communicate with the computer, commanding it to perform tasks and receiving information from it. Understanding the nature of human performance in this common ground is a critically important aspect of HCI. This thesis focuses on interaction occuring at the human/computer interface.

The task of a person interacting with a computer system involves such things as finding information on a CRT screen, reading or otherwise perceiving that information, processing that information to decide on an appropriate response, formulating a plan to execute that response, and executing the response. Surrounding each such local task are the means of learning how to do the task, the detection of errors and recovery from them, and the effects of practice. Thus, HCI includes perceptual processes, motor activities and a preponderance of cognitive skills. Any genuinely useful model of human performance in HCI must address all these areas of the human-computer interface.

People interact with computers in front of a terminal, for sessions ranging from several minutes to several hours, and continuing for days on end. Although the individual sessions may be long, they are broken up into relatively short, loosely coupled activities called *unit tasks* (Card, Moran, & Newell, 1983). For example, a student may spend about fifty hours learning how to program with a Lisp tutor, but these hours are divided into tasks on the order of "find the first element of the list", "increment the counter", etc. This division is especially obvious with a system like a tutor, because the tutor can set these goals for the student and prompt the specific interactions (Anderson & Reiser, 1985). However, sessions with more user-directed systems, like text-editors or spreadsheets, also separate into unit tasks. Thus, the loop of acquiring a unit task, executing it, and acquiring the next unit task forms the organizational unit of interacting at the interface. Understanding the psychology of going around this loop is a central part of modeling the user of a computer system. These short, real-time communications between user and system are at the "interaction level" and are typically less that 30 seconds in duration.

Many different types of psychological research are relevant to the interaction level: controlled experimentation, observations of behavior in real situations, the development of explanatory theories. This thesis is based on the view that a critical component of research relevant to the design of computer systems is the development of predictive theories called "engineering models."

1.2 Engineering Models of Human Performance

Engineering models of human performance are tools in an interface designer's analytic toolbox. Engineering models seek to optimize several criteria that distinguish them from traditional cognitive models: the ability to make zero-parameter predictions, the ability to be taught to and used by practitioners as well as researchers, coverage of total tasks, and approximation. For HCI, any truly useful set of engineering models must cover many of the tasks people perform as they use computers and be integrated enough to explain the interaction between tasks.

1.2.1 Zero-Parameter Predictions

The first criteria for engineering models is the ability to make zero-parameter predictions. Engineering models must be able to make predictions in the absence of a working, or even simulated, system because the predictions are needed early in the design process where they can be used to shape the specifications of the system. Thus, any parameters used to make the predictions must be set during the construction of the model, not when the model is being applied to a new situation. Since a system is not yet in existence, the model cannot require new experiments to be run in the new situation to set the parameters. This does not mean that parameters have to be fixed constants for every situation, only that they must be determinable a priori. Nomographs or tables of parameters covering a wide range of tasks can be used to estimate the parameters and these are created from research done in the design of the model. Such graphs or tables build in the basic psychology so that computer designers can use the models effectively.

1.2.2 Intended End Users - Computer Designers

The second criteria for engineering models is that they must be taught to and used by computer system designers. Since the target users of HCI engineering models are not usually trained psychologists or human factors experts, the user cannot be counted on to bring psychological expertise to the task. Thus, the basic psychology must be built into the models. Nomographs to estimate the parameters of the model in a new situation have already been given as an example of building in this expertise. In addition, the procedures must be clearly defined and representative examples must be present to allow the techniques to be taught. This does not mean that the procedures have to be like recipies in a cook book. After all, system designers learn how to program computers well, and programming is an open-ended task. However, there must be guidelines and rules about what to do in many representative situations so that a style of analysis can be developed in the learner that leads to good predictions.

1.2.3 Coverage

The activities people perform when interacting with a computer are quite varied. As described above, these include reading, searching for information on the CRT screen, processing information relevant to task goals, forming plans, typing, pointing, learning, making and recovering from errors, and a myriad of other activities that overlap and interact with each other. An engineering model of a specific task domain must account for all first-order effects observed in that domain. The approximation inherent in engineering models can not be made by choosing some convenient subset of first-order effects to model, but by the size of the effect (i.e., they must account for all first-order effects, but not necessarily all second-order effects). A useful set of engineering models must do more than focus on each activity in isolation, the models in the set must allow prediction of the overlap and interaction between activities. Coverage is the most difficult criteria for engineering models to satisfy.

1.2.4 Approximation

So far, the criteria discussed put constraints on the type of psychological models that can be useful for engineering purposes. But constraints cannot be imposed on a model without loosening up requirements in other features of the model. The place where engineering models may not be as demanding as more traditional cognitive models is in the area of approximation.

Engineering models in all disciplines of engineering are deliberately approximate. They include just that level of detail necessary to do the design job. For example, when sizing a duct for an air-conditioning unit, fine variations in volume with respect to temperature are ignored if the air can be assumed to be in a certain range of temperature. If this assumption holds, then the equations are very simple, relating the area of the duct to the rate of air passing through it. However, when the same air is traveling through the pipes in the cooling chamber of the air-conditioner, where there is a substantial temperature change from hot to cool, this assumption no longer holds, and many other factors have to be incorporated: relative humidity , compressibility, etc. In the same way, engineering models for HCI must be approximate in nature. It would be impractical to insist on knowing every eye fixation during a session using a relational database, for instance. Such a model would be unnecessarily complex for the design application; it would take a tremendous amount of effort to do those calculations, only experts in interpreting eye-tracking data would be able to do the analysis, and it probably wouldn't provide qualitatively better information for design than a model at a higher level (e.g., the interaction level).

When engineers use approximate models, they do not do so blindly. They have knowledge of a more detailed theory. They know what terms exist in the detailed theory, how sensitive the predictions are to variations in those terms, and, therefore, which terms can be dropped to make calculations tractable without sacrificing the accuracy necessary for the design situation. This can be done because the theory specifies the mechanism of the phenomenon it describes. Similarly, engineering models in HCI must be approximations to the process involved in human behavior, not simply approximations to the behavior. For instance, a model obtained from a factor analysis where the factors have no recognizable correspondence to cognitive mechanisms is of less use than a model where a mechanism is obvious. When the mechanism is explicit, operations can be grouped, averaged over, or ignored, and different levels of approximation can be obtained from one model.

A common engineering guideline is the "80/20 rule", which states that you get 80% of your results from 20% of your effort, and the remaining 20% takes 80% of the effort. Engineering models in HCI should strive for that first 80% of coverage with 20% of the effort. For instance, if a computer system were designed and a prototype built, then the behavior of scores of users could be observed and analyzed in detail and the "truth" would be known about the system. Taking this as 100% knowledge and effort, engineering models in HCI should strive to predict the average user behavior to within 80% accuracy with less than 20% of the effort of prototyping and testing. Thus, less than perfect predictions are acceptable. Engineering models must be detailed enough to attain this level of quantitative prediction.

Although engineering models need not exceed the level of approximation necessary for design purposes, as long as the other criteria are met, such models may be as detailed as their traditional counterparts. In fact, quite detailed and exact models could be incorporated into an easily used computer simulation of the user, or simple models could account for as much performance phenomena as more complex models. Either way, useful engineering models emphasize zero-parameter quantitative predictions, useability, and coverage, sometimes, but not always, at the expense of accuracy.

1.3 Where do Useful Engineering Models Come From?

The requirements for useful engineering models can be met by beginning with a unified theory of human performance that provides the architecture upon which to construct a set of engineering models. The best example of a deliberate attempt to pull together a unified theory of human performance in HCI is presented in Card, Moran and Newell's *The Psychology of Human-Computer Interaction*[1] (1983) (see also Newell and Card, 1985, and Newell and Card, 1986). This volume included a description of a model of the human, called the Model Human Processor (MHP). The MHP is made up of processors and memories that make up the structure of the human perceptual-cognitive-motor system, and principles of operation that govern them.

Although the MHP does provide some coverage over perceptual, cognitive, and motor skills, the components of the model are at such a low level that the requirement of being able to be used by practitioners is not met; too much psychological knowledge is required to combine the components of the MHP into models of complex behavior. Thus, the MHP must be used to create models of complex behavior in HCI task domains before true engineering models are obtained. This has been done in one domain, the expert use of computer command languages, especially text-editing languages (Card, et. al., 1983). The components of the MHP were combined to construct the GOMS family of models, which are intended to be true engineering models, allowing computer designers to make quantitative, approximate, zero-parameter predictions in a range of situations. GOMS models, as applied to text-editing, have received substantial empirical validation (Card, et. al., 1983, Roberts, 1980, Borenstein, 1985). Several researchers have replicated and expanded on the analysis methods used by Card, et. al. (Allen & Scerbo, 1983; Borenstein, 1985; Roberts, 1979). Others have used the components of the MHP and the structure of GOMS to construct engineering models in nearby domains (Kieras & Bovair, 1986; Kieras & Polson, 1985; Olson & Nilsen, in press; Polson & Kieras, 1985; Polson, Muncher, & Engelbeck, 1986; Rosenbloom, 1983; Zeigler, Hoppe, & Fahnrich, 1986).

Considering the criteria for useful engineering models, approximation, zero-parameter predictions, teachability, and coverage, the GOMS models begin to fit the bill. They are approximate in nature, with the grain of analysis varying with the dictates of the task and need for detail. They make zero-parameter predictions using pre-estimated parameters. They have been used by researchers other than the developers of the models, even to the point of being used to make predictions of expert behavior that influenced computer system design decisions (e.g., the number of buttons on the Xerox Star mouse, Card & Moran, 1986). And together they cover a range of human activities: text editing, graphics editing, recall of computer command abbreviations, use of spreadsheets, and within these, performance time and learning. As it stands, GOMS models show substantial promise as useful engineering models for HCI. However, they are still incomplete (in terms of the performance measures they predict, for example) and the extent of their integration remains to be demonstrated.

---

[1] Other possibilities exist, e.g., ACT* (Anderson, 1983), Bailey's review of HCI relevant human performance (Bailey, 1982), and Soar (Laird, Newell, & Rosenbloom, 1987). However, they either have not yet been applied to tasks relevant to HCI, or are not detailed and unified enough to allow engineering models to be constructed from them.

1.4 The Purpose of this Dissertation

1.4.1 What Needs to be Done?

Many models of human performance with computer systems exist only in isolation and in narrow domains. Since different researchers took varied routes in developing engineering models, this appears to be true even among models based on the MHP. Simply designing and running one more experiment to support any one of these models may add to the accumulated knowledge in HCI, but it will not begin to overcome the obstacles of isolation and limited scope. Further research can best contribute to the development of engineering models by extending, consolidating, and making concrete, models already in existence. Any reasonably well developed engineering model could serve as a starting point for such research.

Extension of a model can be accomplished by applying it to phenomena already reported in the literature of HCI, Human Factors, and experimental and cognitive psychology. If the model explains a phenomenon, then the features of that phenomenon that makes it suitable for the model should be identified so the suitability of other phenomenon can be judged against them. If the model does not explain the phenomenon, then the model should be modified to include that phenomenon, or, if such modifications seem impossible, the features of the phenomenon that make it unsuitable for the model should be identified as boundary conditions for the model.

Consolidating existing models begins by realizing that those based on the MHP are not isolated models, however stand-alone they appear to be. They derive from a common base, and are therefore parts of a whole fabric that can be stitched together to produce the most comprehensive and integrated set of models possible. For instance, the Keystroke-Level Model (KLM, a member of the GOMS family) includes an arbitrary, empirically set parameter, the mental operator. This parameter has been shown to be adequate for predicting response times for text-editing tasks (Card, Moran, & Newell, 1983), but inadequate for spreadsheet tasks (Olson & Nilsen, in press). The spreadsheet results could be integrated into the KLM, replacing the single empirically set parameter with three: one for text and data entry, one for scanning, and one for selecting between methods. A better approach would be to integrate visual search theories and a model of stimulus-response compatibility into the KLM, to obtain an independent setting of the mental operator for each situation.

Making engineering models concrete means specifying procedures for a computer system designer to follow to apply the models to a proposed system. If the procedure includes writing algorithms or production systems, then the designer must be taught how to write psychologically valid algorithms or production systems. Either the languages used by designers to apply the model must have the psychology built inescapably into them, or rules and guidelines for using the languages must be established to ensure proper application. In addition, there must be a procedure for estimating any parameters in the model before application to a new computer system.

1.4.2 What this Dissertation Will Do

The purpose of this dissertation, then, is to extend, consolidate and make concrete engineering models based on the MHP. The strong point of existing engineering models is the computation of response times for tasks at the interaction level of HCI. Beginning at the strong point and advancing from there, has a higher likelihood of success than starting from a point of weakness. Therefore, this work starts from the

GOMS family of models and extends their application to two other domains within HCI. In the first domain, immediate behavior, the application of GOMS is taken to a lower level than previous work to allow development of analysis techniques and better estimation of cognitive parameters. These analysis techniques and parameters are then applied to another domain, transcription typing, and used to make zero-parameter a priori predictions of human performance. Thus, this dissertation extends an engineering model, GOMS, to other domains beyond its current state, consolidates it through the sharing of parameters and analysis techniques, and makes it concrete through explicit guidelines and examples of its application.

1.5 The Approach and Organization of this Dissertation

Chapter 2 presents the MHP, its processors, memories and principles of operation, as a base upon which engineering models can be built. It demonstrates the construction of such a model by describing the premises of the GOMS family of models, with a specific example of GOMS applied to the expert execution of command languages (e.g., a text-editing language).

Chapter 3 extends GOMS into the domain of immediate behavior and stimulus-response compatibility. In one experiment, GOMS is applied to a lower level than previous work, to allow prediction of the more detailed operation of the MHP cognitive processor. This analysis, validated by a second experiment and several analyses of results in the stimulus-response literature, allows a narrowing down of a parameter associated with the cognitive processor. This parameter is fed back into the MHP and factored into the construction of engineering models.

Chapter 4 extends GOMS into the domain of transcription typing. The parameters established in the immediate behavior domain are used in this new domain to make quantitative a priori predictions about performance on many typing tasks. The model then covers both domains and is able to predict the results of these two tasks performed concurrently, demonstrating the integration of the model. The tasks and observed performance on them are obtained, not from additional experimentation, but from the typing literature within cognitive psychology and human factors. The typing phenomena deemed robust and important by researchers in the typing field is examined using the model.

Chapter 5 discusses the contribution of this work to the field of HCI. A task analysis of HCI is laid out, existing models in HCI and cognitive psychology are positioned to expose gaps in the HCI knowledge, and this work is shown to fill some of those gaps.

The last chapter presents a brief summary of the dissertation and discusses its contribution to more basic cognitive psychology.

2. The MHP: A Basis for Engineering Models

Unified, integrated engineering models can be created by building them upon a common base. One proposal for a common base, the one adopted in this work, is the Model Human Processor (MHP) described by Card, Moran and Newell in *The Psychology of Human-Computer Interaction* (1983). Since the processors, memories, and principles of operations of the MHP are the building blocks that will be used to construct the engineering models discussed and developed in the remainder of this dissertation, a good understanding of these elements is necessary to assimilate the current work. These elements will be presented here, with the focus on those elements we will use in the development of the engineering models. Greater detail in other areas can be found in the original work.

2.1 Processors and Memories

The general cognitive model of the computer user presented in Chapter 2 of *The Psychology of Human-Computer Interaction* is specified in terms of processors, memories and a few quantitative parameters of each (see Figure 2-1).

There are three separate processors in the MHP: the perceptual processor, the cognitive processor and the motor processor. These processors work serially within themselves, but concurrently with each other, subject to serial limitations imposed by data flow requirements (e.g., the cognitive processor may need information from the perceptual processor before it can proceed). The perceptual processor receives information from the outside world and deposits it into a sensory memory, the most important of which are the Visual Image Store (VIS) and the Auditory Image Store (AIS), where it is held while being symbolically encoded. The Working Memory (WM) receives the symbolically encoded information and the cognitive processor uses that information, together with previously stored information from Long Term Memory (LTM), to make decisions about what to do. The cognitive processor deposits information into WM that initiates action by the motor processor. The motor processor acts on the outside world (e.g., presses keys, responds verbally, etc.).

The processors and memories are described by a few parameters. Card, Moran and Newell derived estimates of these parameters through the distillation of relevant research in cognitive psychology.

The parameter describing each processor is its cycle time, τ. The perceptual processor cycle time, τ_P ranges from 50 to 200 msec, with a typical value of 100 msec. The cognitive processor cycle time, τ_C, ranges from 25 to 170 msec, with a typical value of 70 msec. The motor processor cycle time, τ_M, ranges from 30 to 100 msec, with a typical value of 70 msec.

The parameters describing each memory are its storage capacity in items, μ, the decay time of an item, δ, and the main code type (physical, acoustic, visual, semantic), κ.

For the VIS, its storage capacity, μ_{VIS}, ranges from 7 to 17 letters, with a typical value of 17 letters. The decay time of a letter, δ_{VIS}, ranges from 70 to 1000 msec, with a typical value of 200 msec. The main code type, κ_{VIS}, is physical.

For the AIS, its storage capacity, μ_{AIS}, ranges from 4.4 to 6.2 letters, with a typical value of 5 letters. The decay time of a letter, δ_{AIS}, ranges from 900 to 3500 msec, with a typical value of 1500 msec. The main code type, κ_{AIS}, is physical.

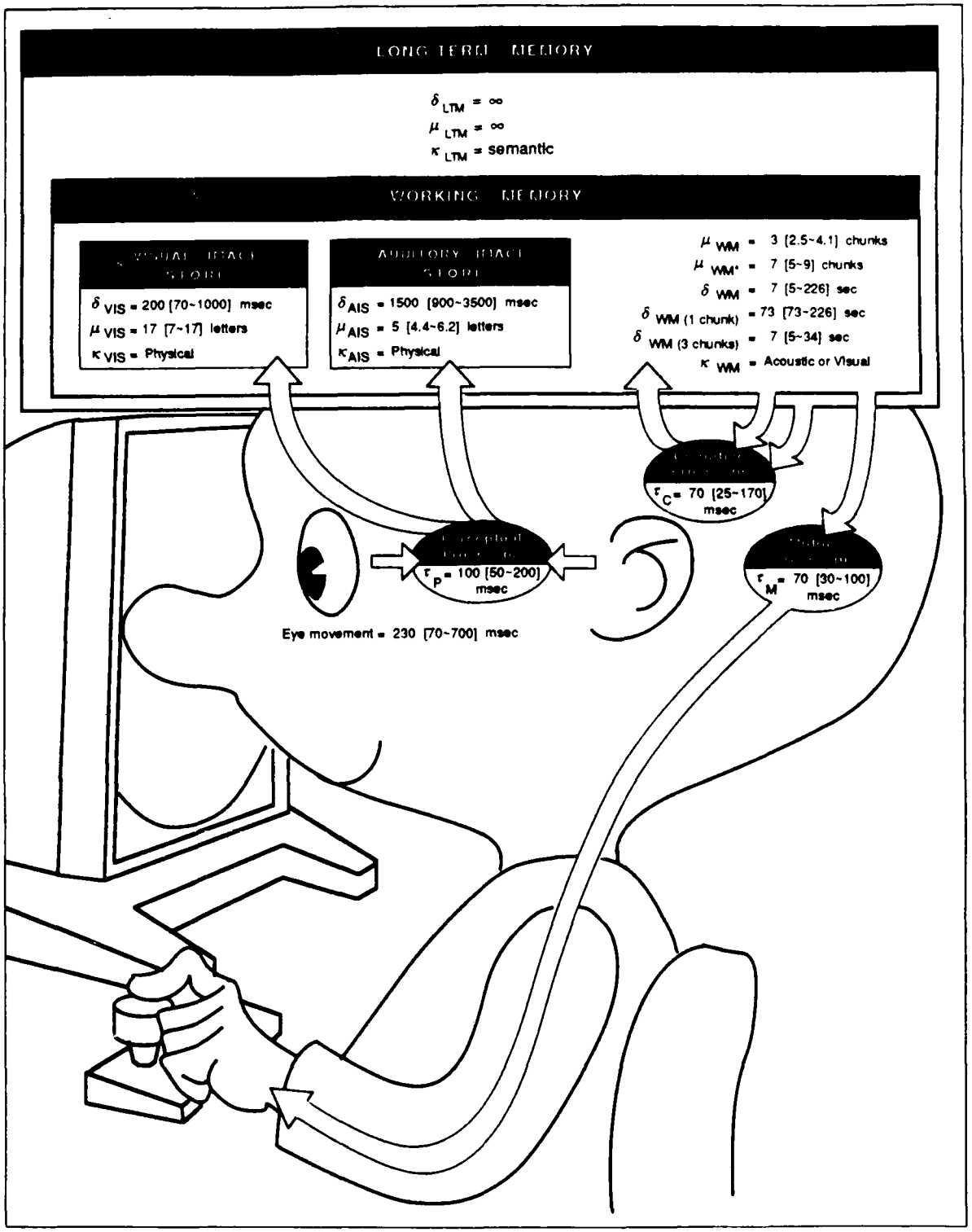


Figure 2-1: The MHP processors and memories
(from Card, Moran, & Newell, 1983, Fig. 2.1)

For WM, its pure storage capacity, μ_{WM}, ranges from 2.5 to 4.1 chunks, with a typical value of 3 chunks. If you allow unpacking of chunks from LTM, then the effective storage capacity of WM, μ_{WM}', ranges from 5 to 9 chunks, with a typical value of 7 chunks. The decay time for WM is dependent on the number of chunks[2]. Overall, the decay time, δ_{WM} ranges from 5 to 226 sec, with a typical value of 7 sec. However, δ_{WM}(1 chunk), ranges from 73 to 226 sec, with a typical value of 73 sec, and δ_{WM}(3 chunks) ranges from 5 to 34 sec, with a typical value of 7 sec. The main code type in WM, κ_{WM}, is acoustic or visual.

For LTM, both the storage capacity, μ_{LTM}, and the decay time, δ_{LTM}, appear to be infinite. The main code type, κ_{LTM}, is semantic.

The processor cycle times are given as fairly wide ranges. They were set to cover the range of times found in the literature without establishing any substructure to the processors. That is, the processors are not specified to the level at which their substructure can be examined, so all operations performed by the processors are assumed to be elementary and the differences in operators are evident only in the wide range of values found when the cycle times are estimated with different tasks.

2.2 Principles of Operation

The processors and memories of the MHP work together under the control of ten principles of operation. These principles cover many aspects of the operations of the three processors, ranging from statements about the cycle times of the processors to a general statement that problem solving takes place in a problem space. They are derived from psychological evidence (e.g., a preponderance of data confirming Fitts's Law) and from a specific model of the psychological world (e.g., P8: the rationality principle). Each principle will be presented with its statement quoted from *The Psychology of Human-Computer Interaction* (Figure 2.2, p. 27). References to the psychological literature supporting these principles can be found in that book.

P0: Recognize-Act Cycle of the Cognitive Processor

On each cycle of the Cognitive Processor, the contents of Working Memory initiate actions associatively linked to them in Long-Term Memory; these actions in turn modify the contents of Working Memory.

P1: Variable Perceptual Processor Rate Principle

The Perceptual Processor cycle time τ_P varies inversely with stimulus intensity.

P2: Encoding Specificity Principle

Specific encoding operations performed on what is perceived determine what is stored, and what is stored determines what retrieval cues are effective in providing access to what is stored.

P3: Discrimination Principle

The difficulty of memory retrieval is determined by the candidates that exist in the memory, relative to the retrieval cues.

P4: Variable Cognitive Processor Rate Principle

The Cognitive Processor cycle time τ_C is shorter when greater effort is induced by increased task demands or information loads; it also diminishes with practice.

---

[2]Although Card, et. al. do not mention it, decay time is approximately independent on each chunk.

P5: Fitts's Law

The time T_{pos} to move the hand to a target of size S which lies a distance D away is given by:

$$T_{pos} = I_M \log_2(D/S + .5),$$

where I_M = 100[70~120] msec/bit.

P6: Power Law of Practice

The time T_n to perform a task on the nth trial follows a power law:

$$T_n = T_1 n^{-\alpha}$$

where α = .4[.2~.6].

P7: Uncertainty Principle

Decision time T increases with uncertainty about the judgement or decision to be made:

$$T = I_C H,$$

where H is the information-theoretic entropy of the decision and I_C = 150[0~157] msec/bit. For n equally probable alternatives (called Hick's law),

$$H = \log_2(n+1).$$

For n alternatives with different probabilities, p_i of occurence,

$$H = \Sigma p_i \log_2(1/p_i + 1).$$

P8: Rationality Principle

A person acts so as to attain his goals through rational action, given the structure of the task and his inputs of information and bounded by limitations on his knowledge and processing ability:

Goals + Tasks + Operators + Inputs +
Knowledge + Process-limits ← Behavior

P9: Problem Space Principle

The rational activity in which people engage to solve a problem can be described in terms of (1) a set of states of knowledge, (2) operators for changing one state into another, (3) constraints on applying operators, and (4) control knowledge for deciding which operator to apply next.

The principles of operation, like the wide ranges for processor cycle times, are an approximation. If the processors were computationally fully specified, then these effects would not be a list of principles, but a direct consequence of the operation of the processors. Card, Moran and Newell were unable to produce a fully integrated model. However, the approximation provided by the principles of operation is still sufficient for many tasks and makes for a much simpler model.

Three illustrations of the application of the principles of operation follow. These examples serve to make the principles concrete and demonstrate that some relevant predictions about human performance can be made even from the application of a single principle.

2.2.1 Using the Encoding Specificity Principle (P2)

Card, Moran and Newell present a clear example of the effects of the Encoding Specificity Principle. A computer user stores a file under the name LIGHT, thinking of LIGHT as the opposite of DARK. Some time later she searches a directory of files, she sees the name LIGHT but thinks of it as the opposite of HEAVY. Since she is using a different set of retrieval cues in the search process than in the storage process, she will not be able to recognize the file.

2.2.2 Using the Discrimination Principle (P3)

An example of the effects of the Discrimination Principle principle is seen when a computer user learns a new text editor with many similar commands. For instance, one version of Emacs uses control-d to delete a character, escape-d to delete a word, control-k to delete the line to the right of "point" (the point between the character on which the cursor is positioned and the character just to the left of it), escape-k to delete from point to the end of the sentence, control-x-rubout to delete from point to the beginning of the sentence, control-w to delete a previously marked region, and a host of other deleting commands. If the learner were to store each of these commands in LTM as DELETE SOMETHING, the retrieval cue, DELETE, would have many associations and any one retrieval would be difficult to make. If, on the other hand, the learner were to use different storage cues, like DELETE UNIT for the "d" commands, KILL MANY UNITS for the "k" commands and WIPE OUT REGION for the "w" commands, then retrieval of any one command would be easier because their are fewer commands associated with each retrieval cue.

2.2.3 Using the Power Law of Practice (P6)

The power law of practice has three practical consequences for the analysis of a highly repetitive task (e.g., typing). First, different skill levels lead to large individual differences. Second, the learning curve is very steep when first learning a task, so the novice goes from totally unpracticed behavior to moderately skilled behavior rather quickly. Third, the curve flattens out, so it is reasonable to talk about relatively stable performance with users of at least moderate skill. This last implication will be drawn upon heavily in the analysis of transcription typing in Chapter 4 of this dissertation.

2.3 Constructing Engineering Models of Complex Tasks

The rudimentary operations performed by the perceptual processor, the cognitive processor, and the motor processor change the user's mental state or cause physical changes in the state of the world. Operators are defined to be sufficiently independent of the task environment that they can be combined to perform many different tasks. They are sufficiently independent of each other that the time to perform each operator in isolation can be used as an estimate of the time to perform that operator within a sequence of operators. These two assumptions of independence allow engineering models of complex user behavior to be constructed from the building blocks of the MHP.

Three examples of engineering models of complex user behavior will be described in some detail in the course of this dissertation. All are members of the GOMS family of models. In this chapter, the premises of GOMS will be described and one example of its application to text editing will be presented. In Chapter 3 the application of GOMS to stimulus-response compatibility (SRC) will be demonstrated. Chapter 4 will build on the SRC model and further extend GOMS to a model of transcription typing.

2.3.1 The GOMS Family of Models

GOMS is a family of models derived from the MHP, the primary influence coming from Principles 8, the Rationality Principle and 9, the Problem Space Principle. GOMS proposes a cognitive structure of the user consisting of (1) the user's Goals, (2) basic Operations that the user could perform to achieve those goals, (3) Methods or combinations of the basic operations, and (4) Selection rules for choosing between alternative methods. This is a family of models for several reasons. First, the basic GOMS structure can be realized within several different control structures. The models described in this dissertation use a hierarchical control structure, but other, more flexible control structures (e.g., production systems) are

equally valid within the GOMS family (Polson & Kieras, 1985). Second, within a single task domain, the grain of analysis can be varied to define basic operations of different duration. Third, when GOMS is applied to different task domains, some task specific assumptions (the sequencing of operators, duration of operators, etc.) are made and the model takes on a sufficiently different surface form that it is convenient to distinguish it from other GOMS models.

2.3.1.1 An Example of a GOMS Model for Text-Editing

When a GOMS model is used to analyze a task, the task is broken down into its goals, operators, methods and selection rules. Before defining these terms, a concrete example is presented to aid in understanding the information-processing definitions given later.

The example (Figure 2-2) describes the task of editing a manuscript with a line editor called POET (described in detail in Card, Moran and Newell, Chapter 3). The highest level goal is to edit the manuscript. This goal is subdivided into smaller goals of completing each unit task (e.g., delete a line, transpose two words, etc.) until there are no more unit tasks to be done in the manuscript. To edit a unit task, the user sets the goals of acquiring (locating and understanding) the task and executing the commands needed to accomplish the unit task. To acquire the unit task, the user may have to go to the next page, or simply get the next task from the page she is already on. To execute the unit task, the goal is set to locate the line on which the change must be made. There are two methods that would accomplish this goal (use a quoted string search, or linefeed to the location); the user must select between them and apply the operators in the chosen method. Then the text must be modified, again involving a selection between methods (substitute or modify commands). Lastly, the user verifies the edit before considering the unit task complete.

GOAL:EDIT-MANUSCRIPT
  GOAL:EDIT-UNIT-TASK *repeat until no more unit tasks*
    GOAL:ACQUIRE-UNIT-TASK
      GET-NEXT-PAGE *if at end of a manuscript page*
      GET-NEXT-TASK
    GOAL:EXECUTE-UNIT-TASK
      GOAL:LOCATE-LINE
        [select: USE-QS-METHOD
        USE-LF-METHOD]
      GOAL:MODIFY-TEXT
        [select: USE-S-COMMAND
        USE-M-COMMAND]
      VERIFY-EDIT

Figure 2-2: An example of a GOMS task analysis

With this example in mind, the formal information-processing definitions of the components of the GOMS model will now be presented.

2.3.1.2 Goals

A goal is a symbolic structure that defines what is to be achieved and determines the set of possible methods that could achieve that state. It also functions as a point of reference to which the system can return if there is a failure or error, where information is stored about what is desired, what methods are available and what has already been tried. For instance, the example contains six goals, **EDIT-MANUSCRIPT**, **EDIT-UNIT-TASK**, **ACQUIRE-UNIT-TASK**, **EXECUTE-UNIT-TASK**, **LOCATE-LINE**, and **MODIFY-TEXT**.

Goals are organized in a goal stack. When a new goal is created, it goes onto the top of the stack and the goals already on the stack are pushed down. When one is completed or abandoned, it pops off the top. This mechanism provides a strict hierarchical control structure.

2.3.1.3 Operators

Operators are elementary acts of the perceptual, motor and cognitive processors. The execution of an operator is the only way to change any aspect of the user's knowledge state or act on the task environment (i.e, the real world). For instance, when the user executes the **GET-NEXT-TASK** operator, she looks down the page until she sees the next place where text needs to be modified and reads the markings describing the necessary modification. This process changes her knowledge state; she now knows what must be done next.

The behavior of the user is the sequence of operators executed by that user. In the example, the sequence of operators in the user's behavior might be

GET-NEXT-TASK
USE-LF-METHOD
USE-S-COMMAND
VERIFY-EDIT

This version of the GOMS model does not admit the possibility of any concurrent operations; behavior is assumed to be the serial execution of operators.

2.3.1.4 Methods

Methods are procedures for accomplishing a task. A method is associated with a goal and contingent on the contents of WM and the state of the task environment. In the example, one method was

GOAL:ACQUIRE-UNIT-TASK
. **GET-NEXT-PAGE** *if at end of a manuscript page*
. **GET-NEXT-TASK**

Depending on the state of the task environment, whether the user is at the end of the page or not, the method will produce the operator sequence **GET-NEXT-PAGE**, **GET-NEXT-TASK** or simply the single operator **GET-NEXT-TASK**.

If methods are guaranteed to succeed (barring errors in execution), as in an editing task, then the task has the character of a cognitive skill. If there is uncertainty about whether a method will advance toward the completion of a goal, then the task is problem-solving.

Methods are procedures that have already been learned and are available for use at performance time, not plans that are created while performing the task. As such, they are one of the major manifestations of skill in accomplishing a task.

2.3.1.5 Selection Rules

When more than one method for accomplishing a goal is available, selection rules must be used to choose between them. Sometimes the task environment dictates which method is appropriate, as when **GET-NEXT-PAGE** must be included in **ACQUIRE-UNIT-TASK** if the user is at the end of the page. Other times the user makes the rule. For example, when the goal is **LOCATE-LINE**, the user might have a selection rule that says to **USE-LF-METHOD** (use linefeed method) if the text in need of modification is less than four lines away from the cursor, but **USE-QS-METHOD** (use a quoted-string method) if it is not. In skilled behavior these selections are made smoothly and quickly.

2.3.2 The Scope and Limitations of GOMS

GOMS models, with their goals, operators, methods and selection rules as defined, can be used to give a complete dynamic description of routine, cognitive skill behavior. Kay and Black (1985) give support to this claim by finding that expert computer users organized commands according to a GOMS structure, but novices did not. Given times associated with operators, the description of expert behavior can provide predictions about the time-course of behavior on a total task. This claim has been verified by empirical evidence presented by Card, et. al. (1983), and others (Roberts, 1979, Singley & Anderson, in press).

As mentioned previously, GOMS models can be constructed at many different levels of analysis. The duration of the operators used defines one dimension along which the grain of analysis can be varied. For instance, Card, Moran and Newell report a study examining nine different GOMS models of text editing at four levels of operator size. The size ranges from an operator that executes an entire unit task to an operator that types a single character on a keyboard. Another dimension for variation of grain size is the treatment of operators at the same level. For instance, the same grain-of-analysis study includes a condition where **LOCATE-LINE** is assigned a single duration and a condition where it is broken into separate cases depending on which method is used to locate the line. Chapters 3 and 4 of this dissertation will report on the performance GOMS models at an even lower level than the Card, Moran and Newell studies, where operators are at the level of cycle times of the three MHP processors.

The version of GOMS presented here, with its hierarchical control structure, provides a good description of error-free performance of a routine cognitive skill. However, it is not sufficient to describe the commission and recovery from errors. There is no mechanism to describe how errors are committed or detected after they are committed. Although the procedures for correcting errors may be routine, there is no interrupt mechanism to break the user out of the subgoal currently at the top of the stack to make the correction. Thus, the hierarchical control structure chosen for the models used in this dissertation should be considered an approximation especially appropriate for error-free, skilled cognitive behavior, and preferred for its great simplicity.

3. A GOMS Model of Immediate Behavior and Stimulus-Response Compatibility

The GOMS model, already presented as an engineering model of expert use of computer command languages (e.g., text-editors), can be used in other domains where its basic components of goals, operators, methods and selection rules characterize the performance of tasks. This chapter demonstrates the use of GOMS in the domain of immediate behavior, i.e., where a stimulus is mapped directly into a response without problem solving or planning. Such behavior is often exhibited in the experimental world with reaction time tasks; the response to a stimulus is well known and initiated immediately upon presentation of the stimulus, without deliberation about what response is appropriate.

Although responses in reaction time tasks are routine and immediate, all such stimulus-response behavior is not of equal difficulty. When stimulus-response pairs in a task result in different performance, the task is said to exhibit stimulus-response compatibility (SRC) effects. Stimulus-response compatibility is a robust psychological phenomenon where the degree of difficulty associated with a response is dependent on the complexity of the relationship between the set of stimuli and their corresponding responses (Fitts & Seeger, 1953). For example, imagine two buttons, one above the other. If these buttons were used to control the summoning of an elevator, the *compatible* configuration would have the up-button *above* the down-button. The *incompatible* configuration would have the up-button *below* the down-button. People would be slower and make more errors using the incompatible configuration than using the compatible configuration. It would not be from lack of knowledge about the correct mapping between the buttons and the commands they represent, because this mapping is evident from the U and D stamped on the buttons, but difficulty in performing the mapping that would cause the degraded performance.

SRC was chosen as the domain within immediate behavior in which to expand GOMS for three reasons. First, it is an important phenomenon in its own right. Many tasks in traditional human factors and human-computer interaction fields are subject to SRC effects. These tasks span a wide range of perceptual, cognitive and motor activities, from hitting a button in response to a light to a more cognitive activity like remembering computer command abbreviations. Second, the differences in performance with different S-R mappings can be gross or subtle, providing a range of behavioral data with which to develop and test models of immediate behavior. Third, some work has already been done relating SRC phenomena to a GOMS engineering model (Rosenbloom, 1983).

3.1 A Brief History of SRC Theory

3.1.1 Early SRC Theory

For almost 30 years after it was experimentally demonstrated (Fitts & Seeger, 1953), no full theory of compatibility was proposed to explain the data that illustrates the phenomenon. Instead, various researchers proposed a piecemeal set of factors that contribute to the phenomenon. Welford (1980) noted this theoretical gap explicitly:

> *Surprisingly, after more than twenty years' research there is no metric for compatibility that transcends particular experimental conditions.*

Rosenbloom (1983) summarized these early theories of SRC, and I follow him closely here.

Deininger and Fitts (Deininger & Fitts, 1955) proposed that since responses must necessarily be

encoded differently from stimuli, SRC is a function of the transformations that must be performed on the stimulus to obtain the response. They went on to propose another factor they felt to be important, whether the pairings of stimuli to responses agreed with population stereotypes, thereby bringing in the effects of prior learning (or inherited tendencies).

Brebner (Brebner, 1973) took the transformation assumption one step further, suggesting that in a multi-step transformation, the use of a single translation process is more compatible than using several different processes. For example, Brebner's hypothesis would imply that it would be easier to do a string of three additions than two additions with a subtraction sandwiched in between.

Several researchers (Filbey and Gazzaniga, 1969, Wickens, Vidulich, & Sandry-Garza, 1984) add another detail to this transformation mechanism by proposing that transformations involving only one hemisphere of the brain are compatible. In addition, Wickens, et. al. propose that the modality of the stimulus (auditory vs. visual) and the response (verbal vs. hand movements in space) also effect the compatibility.

A quantitative explanation of the results of one experiment was proposed by Morin and Grant (1955). There were eight lights and eight response keys below them. A pattern of lights would be presented and the corresponding keys had to be pressed. The mapping between the lights and keys could be changed and ranged from the most intuitively compatible mapping, each key corresponded to the key above it, to a totally random mapping of keys to lights. The rank correlation of each mapping was compared to the mean reaction time for that condition; a linear effect was found for the absolute value of the correlation and a quadratic effect was found for the sign of that correlation. This metric was for this task situation only and no additional experiments were run to test its predictive power.

Shepard (1961) reanalyzed Morin and Grant's results in terms of matrices that specify the patterns of generalizations among the stimuli, P_{SS}, and responses, P_{RR}, (i.e., confusion matrices) and a "permutation matrix", J, specifying the mapping from the stimuli to the responses. The matrix containing the conditional probabilities of the various responses for the various stimuli was P_{SR}, given by the matrix equation,

$$P_{SR} = P_{SS} \cdot J \cdot P_{RR} .$$

The predicted probability of error on each trial was the mean of the conditional probabilities of incorrect responses. These predictions were shown to be a good match to the shape of the response curve, but a poor match to the range of variation observed. This model is applicable to many different SRC tasks, but the confusion matrices must be determined independently for each set of stimuli and responses in a task. Thus, it is a ways away from the goal of zero-parameter predictions.

To explain the results of mixed conditions, where two or more mappings are used (usually one is intuitively compatible and one is not), Smith (1977) proposed a model where excitation is iteratively translated from stimulus to response. In this model, the number of iterations necessary to trigger a response is a function of the excitation level of the stimulus and the excitation criteria of the response. Each S-R pair has an *association time*, reflecting an unspecified function of the compatibility of that pair. The duration of an iteration is proportional to a weighted sum of the association times for all the pairs in the condition (not just of the pair for the current trial). Thus, the iteration time for a particular S-R pair is affected by the association times of the other pairs, (the size of the effect is dependent on the amount of noise in the system). This theory predicts that the time to do a mixed mapping will be between the times to do the respective pure mappings. It also predicts that the addition of more S-R pairs to the condition

will always raise the reaction time for the other pairs (though the effect may be small if the association times of the new pairs are small). Neither of these predictions have been consistantly supported by the literature.

Duncan (1977) proposed that responses might be generated with the use of rules or rule systems. For instance, the incompatible elevator buttons might be remembered as being *opposite* to what you would expect, rather than simply remembering the distinct meanings (top button for down, bottom button for up). In more complex situations, like the mixed conditions Smith studied, Duncan proposed that both the difficulty of these rules, or transformations, and the number of different rules from which to select, influenced compatibility.

Thus, as late as 1980, some factors effecting SRC had been proposed and some attempts had been made to combine these factors in complex situations, but, as Welford noted, no metric had been found to quantitatively predict SRC effects across task situations. In 1983, Rosenbloom proposed such a metric based on an information processing model of human performance.

3.1.2 Rosenbloom's Algorithmic Model of SRC - A GOMS Model

Rosenbloom's *algorithmic model of stimulus-response compatibility* is based on the assumption that "people perform reaction-time tasks by executing *algorithms* (or programs) developed by them for the task. (Rosenbloom, 1983)" Rosenbloom lays out two formulations of this theory. The first is a GOMS model in terms of algorithms. The second goes beyond GOMS, providing a more detailed architecture for modelling reaction time tasks. The first formulation is considered here.

Just as the GOMS model describes human behavior when editing text as a sequence of operators, Rosenbloom's model describes human behavior when executing a reaction time task as a sequence of operators. The operators in Rosenbloom's model are at a lower level than any GOMS model presented by Card, et. al. (1983) by a factor of ten, being on the order of 50 msec. These operators are at the level of the MHP cognitive processor cycle time.

The goals in the tasks studied by Rosenbloom were set up by the experimental situation (e.g., when a light comes on, press the button associated with that light). The methods are the sequence of operators, or algorithms, that the participant performs to accomplish the task. Rosenbloom's theory states that for any reaction-time task, algorithms can be developed either by examining the behavior of people as they perform the task or through task analysis analogous to developing an algorithm for a computer program. Several algorithms may be sufficient to accomplish the task, as is evident in the GOMS text-editing analysis. However, for a reaction-time task, the operators are at such a low level that the algorithms can be verified only by reaction-time data rather than by visible operators as in the text-editing work. Therefore, rather than trying to discern rules for selection between algorithms, every sufficient algorithm (within one order of magnitude of number of operators) is considered and results are averaged over these algorithms.

After algorithms are developed, the number of elementary steps in each algorithm is counted, each step is assigned a cost in milliseconds, and the total time to perform the task is the sum of all the costs. This formulation of the model was totally serial and had only one duration common to every step in the algorithms. Rosenbloom applied this approach to three different reaction-time tasks exhibiting SRC effects: responding with a key-press to the lighting of a light (Duncan, 1977), responding with a joy-stick

movement to the lighting of one or two lights (Fitts & Seeger, 1953), and naming a number when either a numeral or a geometric symbol was presented visually (Morin & Forrin, 1962).

The metric for SRC came from a model of reaction-time tasks, not specifically from an analysis of compatibility. The steps the cognitive procesor must perform to accomplish each reaction-time task are specifed and the difference in the lengths of the algorithms gives the compatibility results. Comparing the number of steps in the algorithms that perform the tasks to the observed performance data, Rosenbloom determined that each step should be assigned a cost of 41 msec. In addition, a different intercept, 300 msec, 238 msec and 389 msec, respectively, should be assigned to each separate experiment, accounting for different perceptual processes in detecting the stimulus, different muscle groups for producing the response, and different amounts of practice. These parameters lead to an excellent fit to the observed data (r^2=0.943). Thus, the algorithmic model of SRC, a GOMS model, was the first model to provide a metric for compatibility (mean response time) that extended to several different SRC tasks.

3.2 Applications of SRC in HCI

Stimulus-response compatibility effects should be expected to appear in many areas of HCI. The most obvious applications are the ones involving spatial SRC, for example, the relationship between function keys and their soft-labels on the CRT screen, and the layout of cursor movement keys. The layout of a specialty keyboard and its relationship to the language it embodies would be another area of concern. As a trivial example, an incompatible layout for a LISP keyboard would have the "(" key to the right of the ")" key.

Landauer, Dumais, Gomez, and Furnas (1982) proposed that SRC effects are present in the assignment of symbolic codes to items in a database. They used Shepard's generalization theorem (Shepard, 1961) to assign alphanumeric codes to animal names based on predetermined similarities between the codes and names. This assignment produced almost 50% faster learning than random assignment of codes; it was a more compatible mapping.

The design of computer languages falls within the realm of SRC. The selection of command names might be influenced by SRC effects. The experiments described in this chapter will demonstrate that SRC effects are present in the selection of command abbreviation rules. The syntax of a language may also be influenced by SRC. For instance, a study of LOGO and BOXER, both LISP derivatives with essentially the same underlying structure, shows novice performance with BOXER to be superior to performance with LOGO (Schweiker & Muthig, ress). The authors attribute this to many factors, not the least of which is that in BOXER "the information is visually structured into *natural* (meaningful) units". This *naturalness* is probably in the mapping between the syntax of the language and the user's mental model of the language.

An analysis technique studying the relationship between a language and the user's mental model of the language, Generalized Transition Networks (GTN), has been proposed by Kieras and Polson (1985). The mapping between the language's GTN and the user's model (embodied in a production system) has implications for the structure of the language itself and/or training and documentation about the language. It is conceivable that this mapping is a version of SRC that can be measured by the GOMS model presented in the remainder of this chapter.

3.3 The GOMS Model of Immediate Behavior

As previously described, GOMS is essentially a model of sequential operations of the processors in the MHP. These processors do the work necessary to accomplish any task, including routine tasks subject to SRC effects. The basic procedure for a GOMS analysis of a task is to break down the task into the gross functions that each of the processors must perform and break these functions down further into elementary actions each processor must take.

Consider the task of generating the correct abbreviation for a computer command (a task that will be discussed in great detail in subsequent sections). In the real-world situation, the command itself is generated by the user when she sets a goal to accomplish a task like deleting a file. Then she has to remember or figure out the abbreviation for "delete". In an experimental situation, the command is given to the participant as a stimulus, specifically, as a command written on a CRT screen. Given the experimental task, the first function that must be performed is to perceive the command, a function of the perceptual processor. Then the abbreviation must be remembered or figured out, a function of the cognitive processor. Lastly, the abbreviation must be typed out on the keyboard, a function of the motor processor. These three functions must be carried out in sequence for the correct abbreviation to be typed in response to a specific command stimulus. These functions might be carried out in different ways, as demonstrated by the next level of task analysis.

Assume, for our level of analysis, that the perception of a word and the execution of muscle movements that hit a key on a keyboard are elementary processes of the perceptual and motor processors, respectively. However, the cognitive function of generating the correct abbreviation might be accomplished in many ways. The person could simply retrieve the abbreviation directly from long term memory, especially if that abbreviation was one she had learned well and used frequently. Alternatively, she could remember the abbreviation through some mnemonic, a process with more steps than a simple retrieval. If the abbreviation obeyed some rule, like deleting all the vowels from the word so "delete" becomes "dlt", and the user knew the rule, she could figure it out from the spelling of the command, examining each letter in the command and typing it only if it was a consonant. This level of task analysis details the steps the cognitive processor would have to go through to perform its function in each of these ways. As an example, the following steps could be used in figuring out the abbreviation (the last method mentioned).

1) Get the spelling of the command from memory.
2) Look at the first letter.
3) If the first letter is a consonant, send a signal to the motor processor to type it.
4) If the first letter is a vowel, ignore it.
5) Look at the next letter.
6) If the next letter is a consonant, send a signal to the motor processor to type it.
7) If the next letter is a vowel, ignore it.
8) Repeat steps 5, 6, and 7 until there are no more letters in the command.

This detailed task analysis yields at least one algorithm for accomplishing a task. Each specific mapping between stimuli and responses in the SRC tasks would produce different sets of algorithms; truncation of a command name to the first two letters would give quite different algorithms than the vowel deletion example above. If each of these algorithms could be expressed as a sequence of steps, each of which was assumed to be an elementary step of one of the three processors, then algorithms could be compared in terms of their lengths, and length could be a predictor of the difficulty of the SRC tasks. If specific values of time could be assigned to the steps of each of the processors, then the total amount of

time it takes to do a task could be predicted. The remainder of this chapter describes exactly this process, beginning with an informal language in which to express the algorithms.

3.3.1 Artless: An Algorithmic Response-Time Language[3]

Algorithms that perform immediate behavior have several features that place them within the GOMS framework.

1. Tasks are accomplished with a sequence of operations of the MHP processors.
2. Each operation has an identifiable duration that is dependent on the processor performing the operation but independent of the context surrounding the operation.
3. Tests in conditionals must be exhaustive and explicit; there is no IF-THEN-ELSE construct.
4. Tests to exit a loop take time only when they succeed.
5. Reasonable algorithms are purely functional (i.e., there are no null operations or empty loops).

These features are embodied in an informal language called Artless. Artless has seven constructs: operators, constants, variables, assignments, blocks, branches, and loops. These constructs will be examined in the light of the computer command abbreviation encoding task.

Before defining the seven constructs, it is useful to introduce a data structure for an ordered list of letters. An ordered list of letters has two functions defined to act upon it: get the first element of the ordered list (First-Letter) and get the next element of the ordered list (Next-Letter). If the list is empty, First-Letter will fail. Next-Letter returns the successor to the member of the list last returned. If the last-returned member of the list is the last member of the list, then Next-Letter will fail. This data structure is used to represent the spelling of commands and abbreviations. When a command or abbreviation is being typed out, once the spelling of the word is gotten, the individual letters come off the ordered list with First-Letter and Next-Letter without cost. If the letters are going to be compared to other letters or tested to see if they are consonants or vowels, etc. then they must be gotten explicitly with a Get-First-Letter or Get-Next-Letter.

Returning to the constructs, the operators do all of the real work in the system. An operator can be thought of as a parameterized black box that performs some computations and optionally returns a value. A typical operator is Get-Stimulus("Command"). The Get-Stimulus operator receives some characterization of what to expect in the stimulus display and it returns a complete description of the object found. In this instance, the object was specified by the constant "Command", so the operator should return the description of a particular stimulus command. Artless is an informal language in that it allows algorithms to be specified that ignore the details of how the concepts "Command" and "complete description of the stimulus object" are represented.

Operators may either succeed or fail. If an operator succeeds, it is guaranteed to have accomplished everything it set out to do. If it fails, it may have been for any number of reasons, such as the lack of needed parameters or because some stimulus situation does not hold. However, the system only knows that the operator failed, not why it failed.

---

[3]The following description of ARTLess is adapted from Rosenbloom (1983) pp. 22-25.

Predicates are a subclass of operators that don't return a value. They are used to test for the truth of some condition. If the predicate succeeds, the condition is true. If it fails, the condition may be false or it may be that the operator did not know how to test the condition in the current situation. A typical predicate is Is-Exception?(Command), which succeeds only if the command is an exception to whatever rule is normally used for constructing its abbreviation.

The assignment construct is used to give values to variables. For example, the following statement assigns the command "delete" to the variable Command, given that the word "delete" appears on the CRT screen.

```
Command ← Get-Stimulus("delete")
```

In order to do a sequence of operators, the block construct is used. The Pascal convention for delineating blocks is used. The block begins with the keyword BEGIN, ends with the keyword END, and has a list of operators between. For example, the following block perceives a command, recalls its two-letter, stores it in an ordered list called Abbreviation, and then types it out.

```
BEGIN
Command ← Get-Stimulus("Command")
Abbreviation ← Get-Abbreviation(Command)
Initiate-Response(Abbreviation[First-Letter])
Execute-Response(Abbreviation[First-Letter])
Initiate-Response(Abbreviation[Next-Letter])
Execute-Response(Abbreviation[Next-Letter])
END
```

Decisions can be made based on the outcome of a predicate (actually, of any operator) by employing a branch. A branch checks for the success of a predicate and does a specific operator or block of operators if the condition is met. For instance, an abbreviation could be retrieved from long-term memory only if it is an exception to the first-two-letters rule.

```
IF-SUCCEEDED Is-Exception?(Command)
  THEN Abbreviation ← Get-Abbreviation(Command)
```

Branches also have an optional ELSE clause that allows some other operator or block to be performed if the predicate failed. However, since the reason why a predicate fails is not known, if the ELSE clause is to execute only if the opposite of the IF clause predicate is true, the ELSE clause must contain an explicit test for the opposite of the IF clause.

```
IF-SUCCEEDED Is-Exception?(Command)
  THEN Abbreviation ← Get-Abbreviation(Command)
  ELSE IF-SUCCEEDED Is-Not-Exception?(Command)
       THEN ...
```

A loop executes an operator, or block of operators, until some predicate becomes true. There must be an explicit test for the predicate at the end of the loop. For instance, if a command were to be typed out in full with a "Return" at the end, and the spelling of that command was stored in an ordered list, then the following block would perform that task.

```
BEGIN
Initiate-Response(Command[First-Letter])
Execute-Response(Command[First-Letter])
REPEAT BEGIN
    Initiate-Response(Command[Next-Letter])
    Execute-Response(Command[Next-Letter])
    END
 UNTIL Done?(Command)
 IF-SUCCEEDED Done?(Command)
  THEN BEGIN
     Initiate-Response("Return")
     Execute-Response("Return")
     END
END
```

As previously shown in the task analysis of generating the abbreviation for a computer command, a variety of algorithms are sufficient for doing the task. In addition, Artless introduces flexibility that introduce more variants of a single general algorithm (e.g. permutations of the order of decisions). Thus, many algorithms may exist for a single task. Each algorithm represents a strategy that a person might use to perform the task (see Newell (1973) and Baron (1978) for discussions of the availability of multiple strategies). Since reaction-time tasks emphasize speed, and indeed the instructions to participants in such experiments often explicitly emphasize speed, it is assumed that people will use a simple (short) algorithm if they can and have had a modest amount of practice. This assumption leads to a simple rule for including algorithms in an analysis: if an algorithm is an order of magnitude longer than the other algorithms in a set that perform a single task, assume that it is not used in practice, and do not include it in the analysis. Even with this constraint, there is often more than one reasonable algorithm. In the absence of data about the relative frequencies of alternative algorithms, make the miminal (Laplacian) mixing assumption: assume the algorithms are equally likely and take their mean behavior.

There is a simple cost model associated with algorithms at this level of detail within the MHP (slightly more complex than the single operator cost used by Rosenbloom): each operator takes one unit of time *associated with the processor that performs it.* Thus, there are three basic operator times: a perceptual operator time, a cognitive operator time, and a motor operator time. These operator times are expected to relate to the processor cycle times discussed by Card, et. al. (1983). All constructs in Artless, other than operators, are free of charge. The REPEAT loop is a special case for counting operators. All the operators within the loop count as they are executed. The predicate in the UNTIL line of the loop does not count, but the predicate outside the loop (re-testing the termination of the loop) does count. This counting procedure mimics the firing of a production in a production system; the production is available but doesn't count until it fires.

3.3.2 Example of an Artless algorithms

The algorithm given previously for figuring out the abbreviation of a command using the vowel deletion rule can be expressed in Artless in the following way.

```
BEGIN                                                          L 1
Stimulus ← Get-Stimulus("Command")                             L 2
Spelling ← Get-Spelling(Stimulus)                              L 3
Initiate-Response(Spelling[First-Letter])                      L 4
Execute-Response(Spelling[First-Letter])                       L 5
Next-Letter ← Get-Next-Letter(Spelling)                        L 6
REPEAT  BEGIN                                                  L 7
        IF- SUCCEEDED Is-Consonant?(Next-Letter)               L 8
           THEN  BEGIN                                         L 9
                 Initiate-Response(Next-Letter)                L10
                 Execute-Response(Next-Letter)                 L11
                 Next-Letter ← Get-Next-Letter(Spelling)       L12
                 END                                           L13
           ELSE IF-  SUCCEEDED Is-Vowel?(Next-Letter)          L14
                     THEN Next-Letter ← Get-Next-Letter(Spelling)  L15
        END                                                    L16
    UNTIL Null?(Next-Letter)                                   L17
    IF- SUCCEEDED Null?(Next-Letter)                           L18
       THEN BEGIN                                              L19
            Initiate-Response("Return")                        L20
            Execute-Response("Return")                         L21
            END                                                L22
END                                                            L23
```

Given this algorithm, L2 is performed by the perceptual processor, L5, L11 and L21 are performed by the motor processor, and the rest are performed by the cognitive processor.

3.3.3 Using Artless to Predict Performance

As presented in Section 3.3.1, a simple cost model is associated with Artless algorithms: each operator is assigned a duration based on which processor performs the operation. Although these operators are expected to relate to the processor cycle times of the MHP given by Card, et. al., it is not clear, a priori, exactly what form this relationship will take. For instance, the perceptual operator used in the Artless algorithms takes what is presented on the CRT screen and converts it to a form from which the spelling of the word can be obtained. This process combines a pure perceptual process with an cognitive encoding process. Artless may not be sufficient for specifying the structure of the encoding process for a complex stimulus, so these processes are not separated, but remain combined in the perceptual operator. Thus the duration of the perceptual operator is expected to be at least the sum of one perceptual processor cycle time and one, or several, cognitive processor cycle time. The durations of the operators are then to be empirically determined.

Once the operator durations are estimated, the response times for different reaction-time tasks can be predicted. Many measures could be chosen to describe the predictive power of the model. Because we are interested in making predictions about human behavior that can be applied to the design of computer systems, the measures about the error between the prediction and the observed behavior are more important than, say, the amount of variance explained by the model. Also, the absolute magnitude of the error is not as important as the proportion of error in a prediction. Therefore, percent error is the right measure to use in evaluating predictions. Consonant with the engineering 80/20 rule, it is hoped that Artless predictions will average at most, about a 20% error with respect to observed behavior.

3.4 A Command Abbreviation Experiment[4]

3.4.1 Purpose of this Experiment

The first experiment was performed for three reasons. First, the experimental task was chosen to demonstrate that SRC effects could be found in a more cognitive task than the choice reaction time tasks usually associated with the SRC phenomenon. The task chosen was within the field of HCI to demonstrate a relevance to design. Second, the experiment was run to demonstrate the explicative power of the GOMS model of SRC. Third, the data from this experiment can be used to estimate the operator durations.

3.4.2 The Task

The task was to recall the abbreviation of a computer command given that you know the function of the command. For example, you want to move *forward* one word in an editor and have to recall that *Escape-f* performs that action. This task, called encoding, can be modelled in an experimental environment by teaching command/abbreviation pairs and asking the participant to type the abbreviation when presented with the command name.

Two abbreviation techniques were chosen for this study. In *vowel-deletion*, the abbreviation is the command name with all the vowels removed (e.g. the abbreviation for "delete" is "dlt"). In *special-character-plus-first-letter* (hereafter called *special-character*), the abbreviation is the first letter of the command name preceded by an arbitrary special character (similar to the abbreviations found in many screen editors, i.e., "<Escape-key>-f" for "forward one word" in Emacs). In addition to the experimental conditions, two control conditions were used. *No-abbreviation*, a maximally compatible task, required the participant to type the full command name in response to the command. *Nonsense*, a minimally compatible task, required the participant to type a nonsense syllable associated with each command.

Qualitatively comparing vowel-deletion and special-character in terms of the complexity of the relationship between the stimuli (the commands) and the responses (the abbreviations), the vowel-deletion relationship appears relatively simple, since the abbreviation can be figured out from the command name in a straightforward manner. The special-character relationship seems more complex, since the arbitrary special character for each command cannot be figured out from the command name itself. However, with the GOMS model of SRC, we can do better than a qualitative evaluation of the complexity. A detailed task analysis and algorithms can provide a metric for comparison.

Beginning with the (intuitively) maximally compatible condition, the no-abbreviation condition, the person would have to perceive the command on the CRT screen, access the spelling of that command (assuming that the person knows how to spell the command and does not have to refer back to the screen for the spelling), send a signal to the motor processor to type out each letter, and type out the letter. An Artless algorithm for that task analysis would be as follows.

---

[4]Earlier analyses of the data from this experiments and the second command abbreviation experiment (section 3.5.1, p. 33) have been published elsewhere (John, Rosenbloom, & Newell, 1985; John & Newell, 1987). Analyses given here are more thorough, giving slightly different (and better) results.

```
BEGIN                                                          L 1
Stimulus ← Get-Stimulus("Command")                             L 2
Spelling ← Get-Spelling(Stimulus)                              L 3
Initiate-Response(Spelling[First-Letter])                      L 4
Execute-Response(Spelling[First-Letter])                       L 5
REPEAT BEGIN                                                   L 6
        Initiate-Response(Spelling[Next-Letter])               L 7
        Execute-Response(Spelling[Next-Letter])                L 8
        END                                                    L 9
 UNTIL Done?(Spelling)                                         L10
 IF-SUCCEEDED Done?(Spelling)                                  L11
   THEN BEGIN                                                  L12
        Initiate-Response("Return")                            L13
        Execute-Response("Return")                             L14
        END                                                    L15
END                                                            L16
```

L2 is a perceptual operator, L5, L8 and L14 are motor operators and the rest are cognitive operators.

An alternative algorithm came from a detailed study of the interkeystroke times actually produced by people performing this task, and involves accessing one syllable of the command at a time. Artless produces the following algorithm for this case.

```
BEGIN                                                          L 1
Stimulus ← Get-Stimulus("Command")                             L 2
Syllable ← Get-First-Syllable(Stimulus)                        L 3
REPEAT BEGIN                                                   L 4
        Spelling ← Get-Spelling(Syllable)                      L 5
        Initiate-Response(Spelling[First-Letter])              L 6
        Execute-Response(Spelling[First-Letter])               L 7
        REPEAT BEGIN                                           L 8
                Initiate-Response(Spelling[Next-Letter])       L 9
                Execute-Response(Spelling[Next-Letter])        L10
                END                                            L11
         UNTIL Done?(Spelling)                                 L12
         IF-SUCCEEDED Done?(Spelling)                          L13
           THEN Syllable ← Get-Next-Syllable(Stimulus)         L14
 UNTIL Null?(Syllable)                                         L15
 IF-SUCCEEDED Null?(Syllable)                                  L16
   THEN BEGIN                                                  L17
        Initiate-Response("Return")                            L18
        Execute-Response("Return")                             L19
        END                                                    L20
END                                                            L21
```

L2 is a perceptual operator, L7, L10 and L19 are motor operators and the rest are cognitive operators.

For the vowel-deletion condition, there are several distinct algorithms. The participant could have the abbreviation so well learned (from previous computer experience) that she could simply retrieve it from long-term memory (using one cognitive operator) and type it out. Or, if the abbreviation was not so well learned, the participant could figure it out by looking at all the letters in the command name and typing out only the consonants. This could be done for the whole word at one time (the example in Section 3.3.2), or by syllables. The letters could be typed out as soon as they are identified as consonants, or all the letters could be examined and then the abbreviation could be typed out in one quick burst. All of the Artless algorithms for these task analyses appear in Appendix II (Volume II).

For the special-character condition, it is assumed that the special characters are not well-learned because they are arbitrary, have no pattern to them, are not used in any real computer system, and the participants have had only a few trials of practice. Therefore, it is unreasonable to expect that they could be retrieved from memory with one cognitive operator. Rather, the participant probably uses some mnemonic or other memory device to aid in the recall. This device, whatever it may be, is probably a composite of many cognitive operators. This composite will be called a retrieval operator and treated as a black box. Thus, an algorithm for generating the special-character condition abbreviations consists of perceiving the command, using a retrieval operator to retrieve the special character, thinking of the spelling of the command, and sending signals to the motor processor to type the special character and the first letter of the command. Then the motor processor types those characters. A second possible algorithm involves memorizing the list of commands and abbreviations in order and searching through them to find the stimulus, thereby accessing the abbreviation (using a variation of the "one is a gun, two is a shoe" mnemonic). Again, a retrieval operator must be used because the ordering of the commands is arbitrary and not extensively practiced. These algorithms are also found in Appendix II (Volume II).

For the nonsense condition, there are two possible algorithms and they parallel the algorithms of the special-character condition. In the first, the abbreviation itself (i.e., the nonsense syllable) is retrieved from long-term memory using a retrieval operator. Then the spelling of the nonsense syllable is accessed and signals are sent to the motor processor to type those letters. The motor processor types them. In the second algorithm, the list of commands is memorized and searched, with the nonsense syllable being associated with the command. A retrieval operator must be used in this algorithm for the same reason as in the parallel algorithm for the special-character condition. These algorithms are also found in Appendix II (Volume II).

Response time is an indicator of SRC effects. It is the measure used most often in the SRC literature, and the theory is in terms of response time. In fact, two response times are of interest. *Initial-response time* is the time to initiate the abbreviation given the command name (measured by the time to hit the first key in a typed response). It reflects the time to perform the mapping between the stimulus and its response, and, thus, reflects SRC. *Execution time* is the time between the initial-response and the typing of the last letter of the abbreviation. If all the characters in an abbreviation are not planned out before the initial character is typed, then execution time includes the time to perform the mapping between the stimulus and the subsequent characters of the response. Thus, execution time may reflect both the difficulty of the response itself and the compatibility of the SR mapping.

Having two times of possible interest complicates a qualitative assessment of the SR relationships. A person might remember the vowel-deletion abbreviation faster, but complete the abbreviation more slowly because there are more letters to type. However, the GOMS model of SRC has no difficulty analyzing this situation and making separate quantitative predictions for each of these response times. The initial-response time is simply the time between the presentation of the stimulus (which is when the perception of the command is assumed to start, i.e., the first step in the algorithm) and the end of the motor operator that types out the first letter of the abbreviation. The execution time is measured by the number of operators needed to complete the typing of the abbreviation. Thus, both times can be accounted for. This experiment will treat both initial-response and execution times.

The numbers of different types of operators (perceptual, cognitive, motor and retrieval) for both of the response times, for each of the algorithms, were counted and averaged together for each of the

conditions. If an algorithm had more than one version because of permutations of decisions, these versions were first averaged together to determine an operator count for that type of algorithm. Then the different types of algorithms were averaged together. The results of this analysis are in Table 3-1.

Table 3-1: Average number of operators for each abbreviation technique.

Abbreviation Technique	Average No. of Operators to First Letter (Initial-Response Time)				Average No. Of Operators between First & Last Letter (Execution Time)			
	Percep.	Cognit.	Retrv.	Motor	Percep.	Cognit.	Retrv.	Motor
No Abbreviation	1	2.50	0	1	0	6.71	0	5.21
Vowel Deletion	1	12.63	0	1	0	9.75	0	2.75
Special Character	1	3.83	1	1	0	1.50	0	1
Nonsense	1	4.25	1	1	0	2	0	2

3.4.3 Method

3.4.3.1 Materials and Apparatus

Twenty-four commands in common use in computer systems were used, ranging in length from five to eight letters (See Volume II, Appendix I). Pairs of commands beginning with the same letter were chosen so that two special characters could be used in the special-character condition. The special characters, "/" and "-", were chosen because they are far enough apart on the keyboard to prohibit typing mistakes, they do not require simultaneous use of two keys (as the "Control" key would), and they are typed with the same finger.

The nonsense abbreviations were three-letter nonsense syllables (0% meaningfulness, from Hilgard, 1953). Twelve nonsense abbreviations were used and no two began with the same letter. A nonsense syllable was only associated with a command that had no letters in common with it. To eliminate any biases due to particular command-nonsense pairs, the linking of each command with a nonsense syllable for each participant was done at random when the experimental session was run.

The experiment was run on the IBM PC/XT and was fully automated, with on-line instructions and automatic data collection.

3.4.3.2 Participants

The participants were 24 Carnegie-Mellon University students taking their first psychology course. All of the participants had computer experience prior to the experiment.

3.4.3.3 Procedure

Each participant was first given a standard typing test to familiarize him or her with the IBM keyboard and to provide an independent measure of typing skill.

Each experimental session had three *conditions*. The first condition was the *no-abbreviation* condition. This condition was always given first to acquaint the participant with the experimental procedure and equipment. It was assumed that this condition, typing a word from copy (in this case, the screen), is a very familiar task to the participants. Thus, performance in this condition would not change over the course of the experiment, nor would practice on this task influence performance in the other conditions. The next two conditions were the *rule-based* (either vowel-deletion or special-character) and the *nonsense* conditions. These conditions were counterbalanced with the rule-based condition first for half the participants and nonsense first for the other half. The participants saw only one of the rule-based conditions because it was thought that learning two different rules might introduce interference, whereas the two control conditions were sufficiently different from the rule-based conditions that they would not. Therefore, all participants saw three out of the four conditions: no-abbreviation, nonsense, and one of the rule-based conditions.

The no-abbreviation condition consisted of a study phase and a testing phase. In the study phase, eight commands (four pairs of commands beginning with the same letter) and their associated abbreviations (the command itself in this condition) were displayed in two columns on the CRT screen for two minutes. In the testing phase, the display was erased and a command appeared in the center of the screen. The participant had to type the abbreviation below the command, then a carriage return to signify the end of the abbreviation. The commands were presented in seven blocks of eight, randomized within each block. The first two presentations of the each command were considered practice and not included in the data analysis.

The rule-based and nonsense conditions consisted of the study phase, a learning phase and then the testing phase. A different list of eight commands and abbreviations was presented for two minutes. This time the abbreviations were nonsense syllables, vowel deletion or special-characters plus first letter. The abbreviation rules were not presented. In the learning phase, a command/abbreviation pair was presented and the participant had to indicate whether the pair belonged together. (The learning phase was not used in the no-abbreviation condition; it was assumed the participants did not need practice in testing the identity of two words.) A drop-out procedure with feedback was used, each command being tested until the person correctly identified the proper abbreviation four consecutive times. After reaching criterion in the learning phase, the participant received the testing phase for the experimental condition. Again, the first two presentations out of seven were not included in the data analysis.

The time (to the nearest millisecond) and the identity of each keystroke were recorded in all phases.

3.4.4 Results

The average typing speed of the participants, as measured by the standard typing test, was 31 gross wpm (or gwpm, i.e., words per minute uncorrected for errors). This indicates the subjects were fairly inexperienced typists.

Since the two control conditions, no-abbreviation and nonsense, were within subjects, along with one of the rule-based conditions, t-tests were done on the non-error trials to demonstrate that the two subject

populations did not differ in their performance in the control conditions. Table 3-2 shows the means and standard deviations for initial-response and execution times for the different subject groups. The results of t-tests (not assuming equal cell variances) gives non-significant *p* values for both the measures in both the control conditions. Therefore, the data will be pooled across all subjects in the control conditions (see Table 3-3).

Table 3-2: Comparison of the two subject groups in the control conditions.

Condition	Dep. Variable	Group	Mean (msec)	S.D. (msec)	n	t	p
No-Abbrev.	Initial-Response	Vowel Del.	830	229	12		
		Spec. Char.	856	234	12	-0.30	0.765
	Execution	Vowel Del.	1381	522	12		
		Spec. Char.	1260	407	12	0.64	0.532
Nonsense	Initial-Response	Vowel Del.	2232	741	12		
		Spec. Char.	2218	510	12	0.05	0.960
	Execution	Vowel Del.	881	636	12		
		Spec. Char.	746	335	12	0.65	0.525

A one-way ANOVA was then performed to compare the performance on non-error trials between conditions: no-abbreviation, vowel-deletion, special-character, and nonsense. It was assumed that the cell variances were not equal and both the Welch and Brown-Forsythe tests[5] show significant difference between the conditions for the intial-response time ($F=38.29$, $p<0.01$ for Welch; $F=33.54$, $p<0.01$ f Brown-Forsythe) and for the execution time ($F=38.29$, $p<0.01$ for Welch; $F=13.66$, $p<0.01$ for Brown-Forsythe). The means are given in Table 3-3.

Table 3-3: Descriptive statistics for experiment 1.

Condition	No-Abbreviation	Vowel-Deletion	Special-Character	Nonsense
Initial-Response Time:				
Mean (msec)	844	1096	1839	2225
S.D. (msec)	227	283	713	622
n	24	12	12	24
Execution Time:				
Mean (msec)	1321	1389	353	813
S.D. (msec)	462	709	124	502
n	24	12	12	24

Thus, with respect to the initial-response time, the four conditions can be ordered as follows. No-abbreviation is the fastest (844 msec), with vowel-deletion 30% slower (1096 msec), special-character 118% slower (1839 msec) and nonsense the slowest (167% slower at 2225 msec). For execution time, the order is special-character the fastest (353 msec), with nonsense 130% slower (813 msec), no-abbreviation 274% slower (1321 msec), and vowel-deletion the slowest (293% slower at 1389 msec).

[5]Run with the BMDP7D statistical program

The number of errors made in the learning phase of the rule-based and nonsense conditions was also collected (vowel-deletion: mean=2.1, sd=2.0; special-character: mean=19.4, sd=16.0; nonsense: mean=22.6, sd=18.7). An ANOVA reveals a significant difference between the conditions ($F=7.24$, $p<0.05$). T-tests show this difference to be between the vowel-deletion condition and the other two conditions ($t=3.72$, $p<0.01$), not between the special-character and nonsense conditions. These data suggest that the assumption that the special-character and nonsense abbreviations were not well-learned is a well founded assumption. Therefore, the set of reasonable algorithms do not include simple (one cognitive operator operation) retrieval from long-term memory. However, the GOMS theory does not yet account for error behavior and errors will not be discussed further.

Table 3-1 (page 27) gives the number of different types of operators for each algorithm for the initial-response and the execution times. An estimate for the duration of each type of operator was obtained by minimizing the least-squares difference between the observed performance and the resulting predictions of intial-response time and execution time. Both the initial-response time data and the execution time data were used to give the maximum number of independent data points. The resulting estimates were that a perceptual operator takes 340 msec, a cognitive operator takes 53 msec, a retrieval operator takes 1249 msec and a motor operator takes 230 msec. Using these results, a plot of theoretical vs. observed initial-response times and execution times (Figure 3-1), has an average absolute percent error of 15.7%.

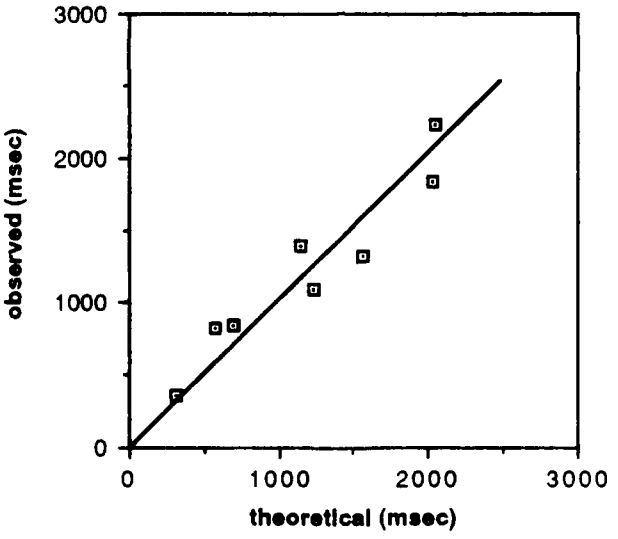


Figure 3-1: Theoretical vs. observed response times.

3.4.5 Discussion

Experiment 1 set out to fulfill three goals: demonstrate SRC effects in a cognitive task within HCI, demonstrate the explicative power of the MHP model of SRC, and produce parameter estimates for future predictions. It clearly accomplished those goals. The SRC effects were demonstrated by the large and significant differences between the inital-response times of the different abbreviation techniques. Initial-response times are most important in demonstrating SRC effects, because initial-response time is traditionally the defining measure for SRC. However, the GOMS model allowed us to go beyond a

traditional intuitive statement of SRC to say quantitatively how complex each of the different abbreviation techniques is, and to identify differences in execution time as well as initial-response time. This is important in design, because comparing these two response times highlights design trade-offs not apparent from intuitive analysis. For example, intuitively, vowel-deletion seems more compatible than special-character, and it is, when only intial response-time is considered. However, when the execution time is also examined, vowel-deletion abbreviations are shown to take much longer to complete than the special-character abbreviations. This execution time difference is due, in part, to the additional letters required in the vowel-deletion abbreviation. This is not the whole story, however, because the no-abbreviation condition requires still more letters and it takes less time to execute than the vowel-deletion condition. Therefore, although vowel-deletion may be an easy rule for people to recognize and learn, it is not a good choice from the point of view of expert performance time. Thus, trade-offs involving novice training time and peak expert performance time must be considered in abbreviation technique selection.

The theoretical calculations leading to predictions within 20% of the observed data verifies the explicative power of the GOMS model of SRC.

Estimates of the operator parameters were generated from the regression analysis, but they should be examined to determine the number of significant digits it is useful to carry over to other SRC experiments and eventually to other task domains. A traditional engineering definition of significant digits or figures is

> As you proceed from left to right in a number..., you will eventually encounter a digit which is doubtful due to the way it was measured or determined...All digits in a number, from left to right, up to and including the doubtful one, are said to be significant figures. (Beakley, Evans, & Keats, 1986, pp. 251-252)

Since the Artless language is a variant of the GOMS model, the operator values estimated from this experiment can be seen as estimates of the MHP processor cycle times. The Card, et. al. estimates of these cycle times shed light on the uncertainty inherent in the parameter estimates.

The cognitive operator, estimated in Experiment 1 to be 53 msec, can be considered an elementary operator of the cognitive processor. Estimates of the MHP cognitive cycle time range from 25-170 msec; 53 msec is well within that range and close to the 70 msec Card et. al. chose for their typical value. However, the range of values cited by Card, et. al. include results from experiments that were not designed to identify an elementary cognitive operator and should be viewed with caution, especially at the high end of the range where composite operators might account for the long time estimates. In addition, the Card, et. al. numbers are the range, not an estimate of variation. Thus, the estimate of variation is inside the range given. Therefore, the cognitive operator is uncertain in the ten's digit and will be taken to be 50 msec for the remainder of the analyses in this dissertation.

The perceptual operator is estimated to be 340 msec by regression. This operator codes the words on the CRT screen as well as perceives them. Thus, it is a combination of a perceptual processor cycle time (50~200 msec, 100 msec typical value) and at least one cognitive processor cycle time (50 msec). Again, the numbers given by Card, et. al. are the range, and the variation is inside that, making the perceptual processor cycle time uncertain in the ten's digit. A sum of values significant to the ten's digit is also significant to the ten's digit. Therefore, the perceptual operator will be taken to be 340 msec (where the ones digit is not significant).

The motor operator is estimated to be 230 msec, high for the range of motor processor cycle times (30~100 msec, typical value 70 msec). However, the participants in this experiment were not very skilled typists (average 31 gross wpm). This could mean some of the activities contributing to the high average

motor operator estimate might be hunting for the correct key and pecking at it, as well as the more straight-forward motor activities used to estimate the motor processor cycle time. In the absence of data illuminating the motor activities actually involved, the operator duration estimate will remain a black-box estimate of the time to hit a key on a keyboard for a population with a 31 gwpm typing speed. Since the perceptual processor cycle time, the motor processor cycle time and the cognitive processor cycle time are significant in the tens digit, a sum of several of those operators would also be significant in the tens digit. Therefore the parameter estimate carried along will be 230 msec (where the ones digit is not significant).

A retrieval operator, estimated at 1249 msec, performs the recall of an arbitrary connection between two items. This is essentially an arbitrary paired-associate task, though done internally (hence without perception and motor components). A few studies of simple paired-associates tasks report latencies, reporting means between 1615 and 2020 msec (Brown & Huda, 1961; (Williams, 1962); Millward, 1964). These times of course are for a complete task and the retrieval operator is just one component. The simplest algorithm for a paired-associate recall task would have one perception operator, one retrieval operator, and one motor operator. Thus, subtracting 570 msec (the duration of one perceptual operator plus one motor operator) from the latency data gives an estimate of the retrieval operator in the range 1045 to 1450 msec. This brackets the retrieval operator value of 1249 msec and casts doubt on the hundreds digit. In accord with this analysis, the retrieval operator reflects many individualized methods for accomplishing the retrieval. These different methods make the retrieval operator more than just the sum of a constant number of cognitive operators; it increases the variance of the resulting estimate. Thus, only the first two significant digits should be carried along for the retrieval operator duration, making that estimate 1200 msec.

This analysis in terms of paired-associate recall makes the value of the retrieval operator reasonable, but it does nothing to explain its magnitude. Much work on mediation in paired-associates learning indicates that this operator is composite (Jensen & Rohwer, 1963; Runquist & Farley, 1964; Montague, Adams & Kiess, 1966). For a particularly detailed view of how natural language mediation might work see Prytulak (1971). Support for the interpretation of a retrieval operator as a composite of several cognitive operators also comes from protocol data collected in the next experiment. A subject remembered that the abbreviation for the command "apply" was "al" because "apply" reminded him of applying for a job and that reminded him of his friend, Al, who had just done so. Decomposing the retrieval operator is an important task, but the function of the composite operator is sufficiently clear that it can be employed successfully in constructing task algorithms.

The new operator duration estimates applied to the average operator counts for the conditions in Experiment 1, yield predictions with an average absolute error of 15.8%. This result is essentially the same as that obtained with the original, unrounded estimates, which is to be expected, because the extra digits simply add random error to the predictions. The rounded operator duration estimates, (340 msec for a perceptual operator that reads a word, 50 msec for a cognitive operator, 1200 for a retrieval operator and 230 for a motor operator that finds and types a character at a rate of 31 gwpm) will be carried to the next section in the second step toward an engineering model with predictive power: making a priori predictions for a new tasks.

3.5 Making Predictions of Response Time Performance

After estimating operator duration parameters, those parameters can be used to predict response time performance on other immediate behavior tasks. Four such tasks will be explored: a second command abbreviation recall task with different abbreviation techniques, two spatial SRC tasks, and a symbolic SRC task. The command abbreviation task analysis predicts performance measured in a new experiment. The other three tasks predict performance reported in the SRC literature and are the same tasks examined by Rosenbloom (1983).

3.5.1 A Second Command Abbreviation Experiment

3.5.1.1 The Task

Experiment 2 studied two different abbreviation techniques. Truncation to the minimum number of letters to distinguish between commands (hereafter, *MTD* for *minimum to distinguish*). MTD maps the commands *compile, create, define* and *delete* into *co, cr, def,* and *del,* respectively. Truncation to two letters, with exceptions to distinguish between ambiguous commands (hereafter, *2-letter*) maps the same commands into *co, cr,* (as does MTD) and *de* and *dl,* respectively. The rule for the exception, when two commands cannot be distinguished by their first two letters, is to use the first distinguishing letter as the second letter in the abbreviation, for the command second in alphabetical order.

Predictions of response time for a new SRC task can be made by doing a detailed analysis of the task, writing algorithms, counting the operators in the algorithm and multiplying those counts by the previously determined parameters.

Since many algorithms are possible for a task, different people can be expected to use different algorithms. In general it is not possible to predict a priori which algorithms will be used by any given individual. The simple solution, used successfully in experiment 1 and adopted here, is to generate the population of different algorithms for the task and then average them together. If the major algorithms are generated and the population is relatively large, then the absence of rare algorithms will make little difference. In the absence of better information all distinctly different algorithms are weighted equally.

Algorithms for the MTD and 2-letter encoding task were generated according to a series of five design choices Figure 3-2. The first choice is to recall the abbreviation for a command directly, as if it were a paired associate (the *recall-abbreviation* option); to recall a feature of the abbreviation, e.g., how many letters it contains (*recall-feature*); or to figure out the abbreviation knowing the rules given above (*figure-out*).

A second choice is between treating all commands the same way (*single-method*) or using different methods for different cases (combined). A third choice arises in how to do the combining. If only a few commands do not fit the primary rule (the abbreviation being the first two letters), then those few commands can be marked as exceptions and the abbreviations can be either recalled or figured-out whenever an exception is encountered (*by-exception*). Alternatively, if approximately an equal number of commands have different types of abbreviations (e.g., first-two-letters, first-three-letters and first-four-letters in the MTD condition), then categories can be set up to reflect these types and each command can be marked with its category; once the category is retrieved, the abbreviation can be either recalled or figured out (*by-category*).

Choice 1	Choice 2	Choice 3	Choice 4	Choice 5
Recall-Abbrev.	One-Method			Type
				Wait
	Combined	By-exception	Marked-by-word	Type
				Wait
			Marked-by-letter	Type
				Wait
		By-category	Marked-by-word	Type
				Wait
			Marked-by-letter	Type
				Wait
Recall-Feature	One-Method			Type
				Wait
	Combined	By-exception	Marked-by-word	Type
				Wait
			Marked-by-letter	Type
				Wait
		By-category	Marked-by-word	Type
				Wait
			Marked-by-letter	Type
				Wait
Figure-Out	One-Method			Type
				Wait
	Combined	By-exception	Marked-by-word	Type
				Wait
			Marked-by-letter	Type
				Wait
		By-category	Marked-by-word	Type
				Wait
			Marked-by-letter	Type
				Wait

Figure 3-2: Choices for generating abbreviation methods.

A fourth choice arises because exceptions and categories can be associated with either of two features of the commands: the whole word (*mark-by-word*) or the first letter (*mark-by-letter*). For example, in the 2-letter condition, if a by-exception method is used with the whole word, delete will be flagged as an exception, but not define. However, if the first letter is used to mark the exception, both delete and define will be flagged.

A fifth choice arises when the response begins. The participant can type as soon as a letter of the abbreviation is generated (*type*), or the participant can wait until the entire abbreviation is generated and then type (*wait*).

Each one of these different methods for generating the abbreviations can be embodied in an algorithm written in Artless. However, when the details are worked out, not all of these possible methods are distinct algorithms. For instance, 2-letter by-category, marked-by-letter is not unique because the two commands starting with the same letter are in different abbreviation categories (e.g., define would be in a first-two-letters category and delete would be in a first-and-third-letter category), which is equivalent to mark-by-word. Different versions of these algorithms arise because every time a test is made in an algorithm, a version of that algorithm could be constructed that permutes the order of the tests. In all, there are 71 algorithms for the MTD condition and 55 algorithms for the 2-letter condition (Volume II, Appendix III).

Predictions are made by getting an average number of operators for the different abbreviation tasks.[6] These averages are combined with the parameter estimates determined in experiment 1 with the following equation:

$$T_{predicted} = 340^{*}n_{perceptual} + 50^{*}n_{cognitive} + 1200^{*}n_{retrieval} + 230^{*}n_{motor}$$

The predictions will be rounded to the nearest ten msec. The predictions for the two new abbreviation techniques and for a no-abbreviation control condition (derived from the algorithms for that task developed in the Experiment 1), are found in Table 3-4.

3.5.1.2 Method

The second experiment was essentially identical to the first experiment. Twenty-four students, with backgrounds simular to those in experiment 1, participated in the experiment. The same commands were used, but eight new commands were added (four sets of two beginning with the same first letter) to give more command pairs with three- and four-letter abbreviations (or first-and-third- and first-and-fourth-letter abbreviations) (See Volume II, Appendix I). Only one control condition, the no-abbreviation condition, was used, and participants saw twelve commands in each condition, rather than eight. Again, the no-abbreviation condition was given first to familiarize the participants with the procedure and the keyboard. Again, a between subject design was used for the rule-based condition to eliminate possible interference between abbreviation rules. Thus, each participant had two conditions, first the control and then one of the rule-based conditions.

As in experiment 1, each participant was first given a standard typing test, then the abbreviation conditions. At the end of experiment 2, the participant was asked to write a description of how she or he remembered the abbreviations in the experiment.

---

[6]As in Experiment 1, versions of the algorithms created by permuting the order of decisions were averaged together first to provide an estimate for that type of algorithm. Then the different types of algorithms were given equal weight.

Table 3-4: Predictions of response times.

Condition	Type of RT	Number of Operators				Prediction (msec)
		Perception	Cognitive	Retrieval	Motor	
No Abbreviation	initial response	1	2.50	0	1	700
No Abbreviation	execution	0	6.81	0	5.31	1560
Minimum-to Distinguish	initial response	1	4.31	0.30	1	1150
Minimum-to Distinguish	execution	0	3.88	0.06	2	730
2-Letter	initial response	1	4.27	0.28	1	1120
2-Letter	execution	0	3.27	0.10	1	510

3.5.1.3 Results

The average typing speed of these participants was 26 gross wpm, indicating again relatively unskilled typists.

Table 3-5 presents the observed response times and the theoretical predictions for the initial-response times and execution times generated from the number of operators in the algorithms (see Table 3-4, page 36) and the previously determined operator parameters. Figure 3-3 plots the predicted times vs. observed times, with an average absolute percent error of 27.2%.

Table 3-5: Observations and predictions of response times.

Condition	Type of RT	Observed (msec)	SD (msec)	Prediction (msec)	% Error
No Abbreviation	initial	955	397	700	36.7
No Abbreviation	execution	1554	799	1560	- 0.4
MTD	initial	1064	375	1150	- 8.1
MTD	execution	525	389	730	-39.0
2-Letter	initial	1032	442	1120	- 8.5
2-Letter	execution	299	216	510	-70.6
				Average absolute % error	27.2

The methods used in the predictive algorithms are rational methods, since they suffice to perform the task and do so within a language that conforms to the premises of the MHP. However, not all of the methods need occur in human behavior or occur with equal frequency. The protocols taken after the experiments shed some light on this. They contained positive indications that at least some participants chose each option of the choices for recall-abbreviation vs. recall-feature vs. figure-out, one-method vs.

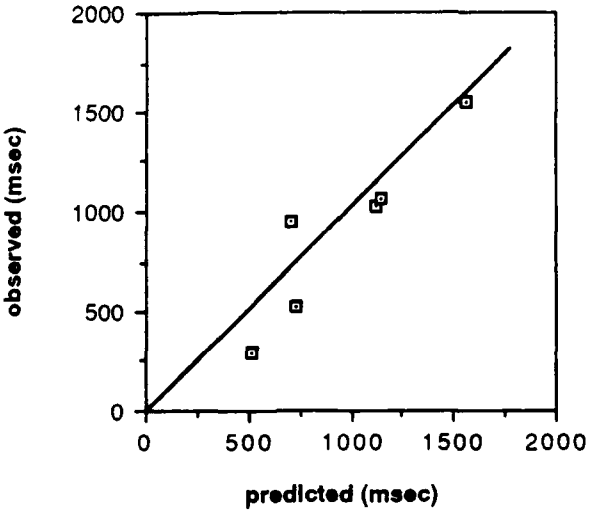


Figure 3-3: Predicted vs. observed response times for experiment 2.

combined, by-exception vs. by-category, and marked-by-word vs. marked-by-letter. Differences between type and wait were not found in the protocols and probably would not be evoked by the instructions given for them. Also, the protocols gave no indication of additional characteristics of methods that were not considered in the *a priori* task analysis.

3.5.2 The Fitts & Seeger Task

The immediate behavior task studied by Fitts and Seeger (1953) was for the participant to move a stylus in response to the lighting of a lamp. Fitts and Seeger crossed three stimulus conditions with three response conditions. Only two stimulus and response conditions will be examined here because the C stimulus and response conditions involve motor movement complexities that have not yet been incorporated into the model. The details of the task are defined by the apparatus used:

> Stimulus Set *A* and Response Set *A* each consisted of eight permutations of direction from a central reference point. the stimulus panel contained eight lights forming the outline of a circle. The response panel contained eight pathways radiating from a central point like the spokes of a wheel. Each light and each pathway were separated from their neighbors by an angle of 45°...
>
> Stimulus panel *B* consisted of four lights separated by 90°. It provided four single-light positions, plus the four two-light combinations formed by adjacent pairs of lights. Response panel *B* consisted of four pathways originating at a central point and separated by 90° intervals. Each path, as seen in [Figure 3-4], branched in a T and permitted *S*s to move from the center point to one of eight terminal positions. The four corner points of the response panel could be reached by two alternative pathways: for example, the upper-right corner could be reached either by a right-up sequence or an up-right sequence. The *S*s were told that these were equivalent responses...
>
> At the center of each response panel was a 1/8-in. diameter metal disc, surrounded by a thin nonconductive ring. Reaction time was measured as the time taken by *S* to move the stylus off this metal button.

Algorithms can be written in terms of perceptual, cognitive and motor operators that accomplish these tasks. However, the perceptual and motor operators are different from those in the command abbreviation

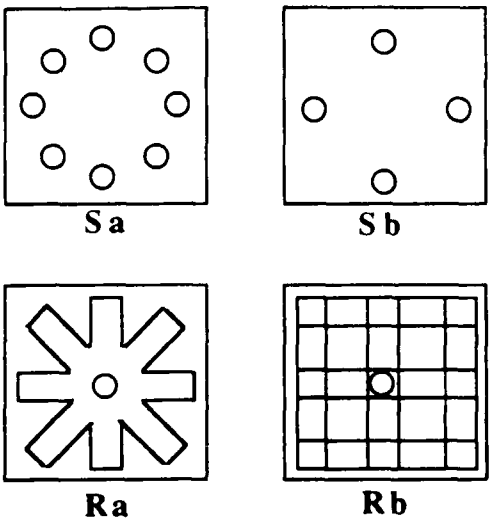


Figure 3-4: The two stimulus panels and the two response panels used in this analysis, from Fitts & Seeger, 1953, p. 202.

experiments because the perceptual and motor tasks are different. In order to make zero-parameter, a priori predictions of the behavior in these tasks, some independent estimate of appropriate perceptual and motor operator durations must be made.

In the absence of data about the perceptual and motor operators for this task, the typical value of the MHP perceptual and motor processor cycle times will be used (100 msec and 70 msec, respectively). This estimate alone, is sufficient for the motor response, because that response has been assumed to be a function purely of the motor processor all through this chapter.

The perceptual process is more complex, involving both pure perception and encoding into meaningful symbols. Assuming that the estimate of the perceptual processor cycle time is an estimate of the pure perceptual portion of this process, cognitive operators must be included in the algorithms to account for the encoding process. With simple stimuli, like lighting one out of eight lights, the encoding process could be a simple discrimination process, subject to information theory analyses. It will be assumed that for this stimulus, and for the stimuli in the next two analyses as well, encoding is accomplished with one cognitive operator for each bit of information necessary to discriminate between stimuli. Thus, with eight possible stimuli, three bits of information are needed and three cognitive operators are included in the algorithms to perform the encoding.

The algorithms for the different S-R conditions in this experiment can be found in Appendix IV (Volume II). The predictions arising from these algorithms combined with the operator duration estimates (perceptual=100 msec, cognitive=50 msec, motor=70 msec), along with the actual observed response times, are presented in Table 3-6 and Figure 3-5. These predictions have an average absolute percent error of 16.1%

Table 3-6: Predicted and observed response times for the Fitts & Seeger expenment

Condition	Observed (msec)	Number of Operators Perception	Cognitive	Motor	Prediction (msec)	% Error
Sa-Ra	390	1.00	5.00	1.00	420	- 7.7
Sa-Rb	430	1.00	7.00	1.00	520	-20.9
Sb-Ra	450	1.50	7 50	1.00	600	-33 3
Sb-Rb	410	1.00	5.00	1.00	420	- 2.4
					Average absolute % error	16.1

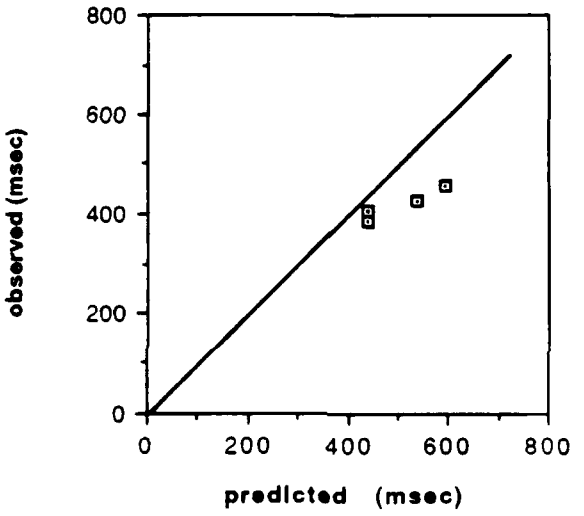


Figure 3-5: Predicted vs. observed response time for the Fitts & Seeger experiment.

3.5.3 The Duncan Task

Duncan (1977) studied a immediate behavior task where the participant had to hit a button in response to the lighting of a line on an oscilloscope. His apparatus and task were as follows:

> The stimuli were four vertical lines, arranged in a horizontal row on an oscilloscope...(Fig. 1) [here, Figure 3-6]...the stimuli have been numbered 1 to 4 from left to right. Stimuli 1 and 4 will be termed "Outer" and Stimuli 2 and 3 "Inner."
>
> Responses were made with four keys arranged to lie under the fore- and middle fingers of each hand...the responses have been labeled from A to D from left to right.
>
> Figure 1 [Figure 3-6] illustrates the four S-R mapping rules employed. Arrows connect stimuli to their appropriate responses.
>
> ...the following terminology will be employed. S-R pairs will be termed either "Corresponding" or "Opposite". An example of a Corresponding pair is "1-A," and of an Opposite pair "1-D." Mappings which contain only one type of pair will be termed "Pure," while those which contain both will be termed "Mixed."
>
> ...Each subject served in...one of the four experimental conditions. At the start the S-R mapping was described by numbering the stimuli from 1 to 4, and showing which key corresponded to each number.

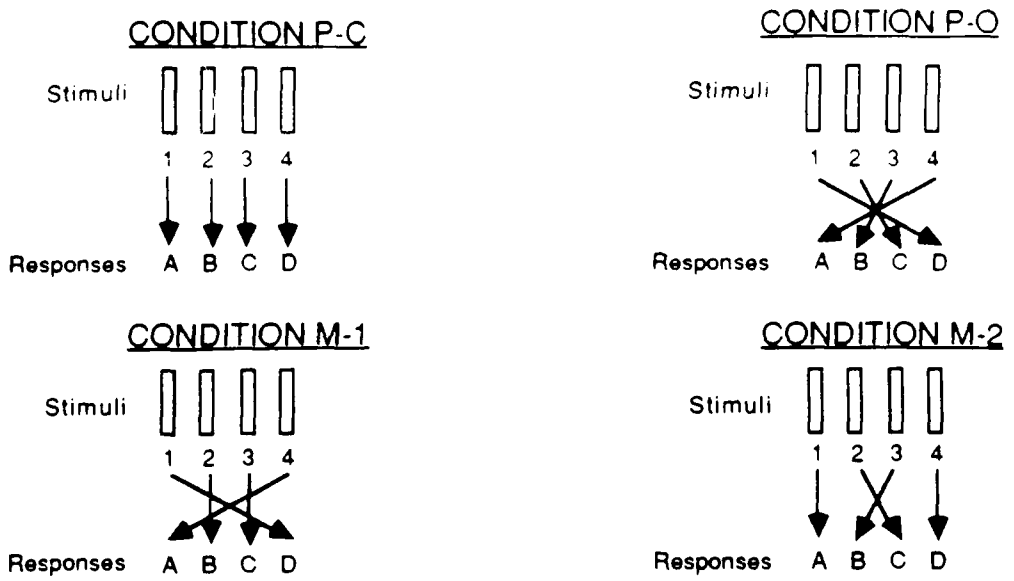


Figure 3-6: S-R mappings in the four experimental conditions from Duncan, 1977, p. 51.

As in the algorithms for the Fitts and Seeger tasks, the perceptual process is broken down into one pure perceptual operator of 100 msec, and one cognitive operator for each bit of information necessary to accomplish the task. In this case, the participant must discriminate one of four possible vertical lines, requiring two bits of information. Thus, there are two cognitive operators to perform the discrimination in each of the algorithms.

The motor operator will again be assumed to take 70 msec, even though the motor response of pressing a key is different from the stylus movement in the Fitts and Seeger experiment. This typical value of the motor processor cycle time will be used as a best estimate whenever there is no data to support a different estimate.

The algorithms for this experiment can be found in Appendix V (Volume II). The predictions dictions and actual observed response times are presented in Table 3-7 and Figure 3-7. These predictions have an average absolute percent error of 8.3%

Table 3-7: Predicted and observed response times for the Duncan experiment.

Condition	Observed (msec)	Number of Operators			Prediction (msec)	% Error
		Perception	Cognitive	Motor		
Pure-Corresponding	411	1.00	4.00	1.00	370	10.0
Pure-Opposite	461	1.00	5.00	1.00	420	8.9
Mixed-Corresponding	485	1.00	5.50	1.00	450	7.2
Mixed-Opposite	539	1.00	6.50	1.00	500	7.2
				Average absolute % error		8.3

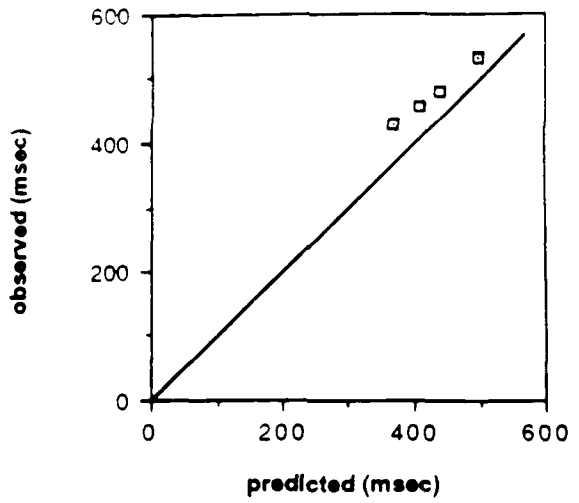


Figure 3-7: Predicted vs. observed response time for the Duncan experiment.

3.5.4 The Morin & Forrin Task

The task used by Morin and Forrin (1962) was for the participant to respond verbally with a numeral, when presented visually with a numeral or a symbol (cross, square, circle or triangle). Figure 3-8 shows the five experimental conditions. Morin and Forrin separated the experimental conditions into Experiment A and Experiment B for purely expository purposes. Experiment A looked at naming numerals without context (condition I), with a context of a similar task (condition IV), and with a context of a different task (condition IIIA). Experiment B looked at responding to a symbol with a numeral without context (condition II), with a context of a different task (condition IIIB), and with a context of similar task. The first element in each pair served as the stimulus and the second element as the response. For example, in condition IIIB, a cross would be shown on a screen and the subject would respond by saying "four" (a "critical pair"), or a "2" would be presented and the subject would say "two" (a "context pair"). Response times were analyzed in trials of the critical pairs only, not for the context trials in conditions where context was present.

In this experiment, the discrimination of the visual stimuli is different depending on the experimental condition. In conditions I and II, there are only two possible stimuli and thus only one cognitive operator is needed, corresponding to one bit of information. In conditions IV and V, four different stimuli are possible, requiring two cognitive operators to perform the encoding. In conditions IIIA and IIIB, two cognitive operators are also needed: one to discriminate between numeral and symbol, and one to discriminate between either the two numerals or the two symbols.

Again, there is no data to point toward an estimate of the duration of a verbal response, so the typical value of the motor processor cycle time, 70 msec, will be used as the estimate for the motor operator duration.

The algorithms for this experiment can be found in Appendix VI (Volume II). The predictions and actual

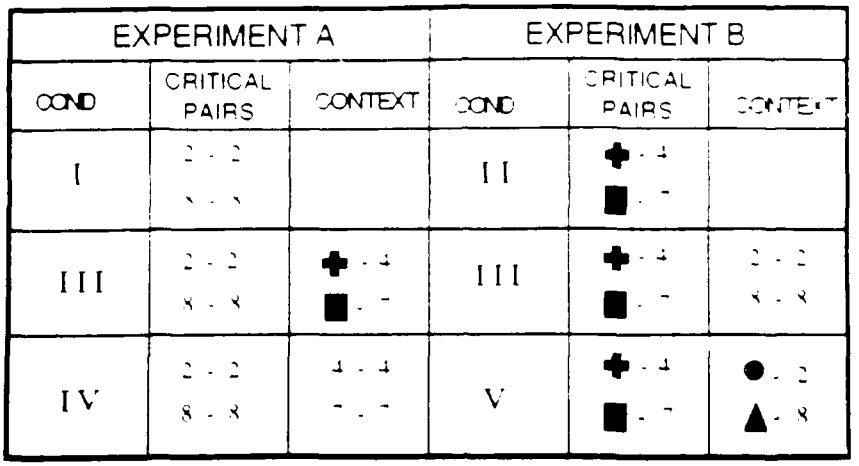


Figure 3-8: Schematic diagram of the experimental conditions from Morin & Forrin, 1962, p. 138.

observed response times are presented in Table 3-8 and Figure 3-9. These predictions have an average absolute percent error of 23.4%

Table 3-8: Predicted and observed response times for the Morin & Forrin experiment.

Condition	Observed (msec)	Perception	Number of Operators Cognitive	Motor	Prediction (msec)	% Error
I	520	1.00	3.00	1.00	320	38.5
II	590	1.00	5.50	1.00	450	23.7
IIIa	600	1.00	6.50	1.00	500	16.7
IIIb	710	1.00	9.00	1.00	620	12.7
IV	490	1.00	4.00	1.00	370	24.5
V	720	1.00	7.50	1.00	550	24.3
					Average absolute % error	23.4

3.6 General Discussion of the GOMS Model of Immediate Behavior and SRC

The GOMS model presented in this chapter is a model of how people perform immediate behavior tasks. Zero-parameter predictions of response time for four different types of tasks have been made with average absolute percent errors ranging from 8.3% to 27.2%, with an overall average of 18.8% (see Figure 3-10). This is in keeping with the goal of an engineering model to be able to make zero-parameter predictions averaging within 20% of actual performance. The predictions of different response times to different stimulus-response mappings allow prediction of SRC effects as well.

The analysis necessary to obtain good response time predictions extends the GOMS model so a lower level of detail is added to the MHP cognitive processor. The cognitive operators are still anonymous, that is, they are only informally specified and no distinction is made between operators on the basis of duration, but the structure of some cognitive processes (e.g., remembering the mapping between lights and buttons) is partially defined. This is significant because this process might have been presented as a MHP principle of operation (e.g., P10: Principle of SRC. Simpler S-R mappings require less time to

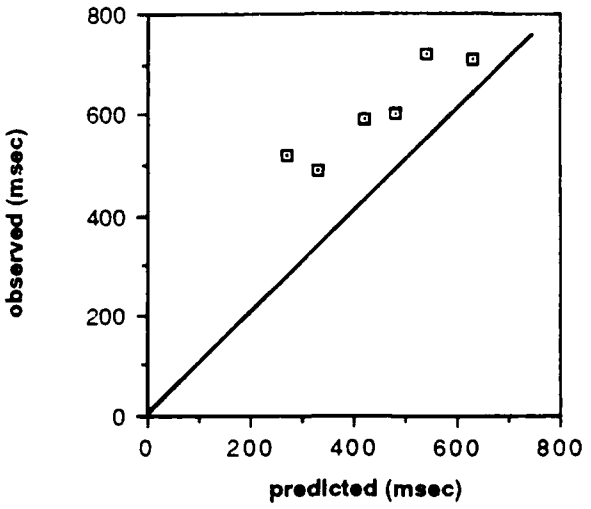


Figure 3-9: Predicted vs. observed response time for the Morin & Forrin experiment.

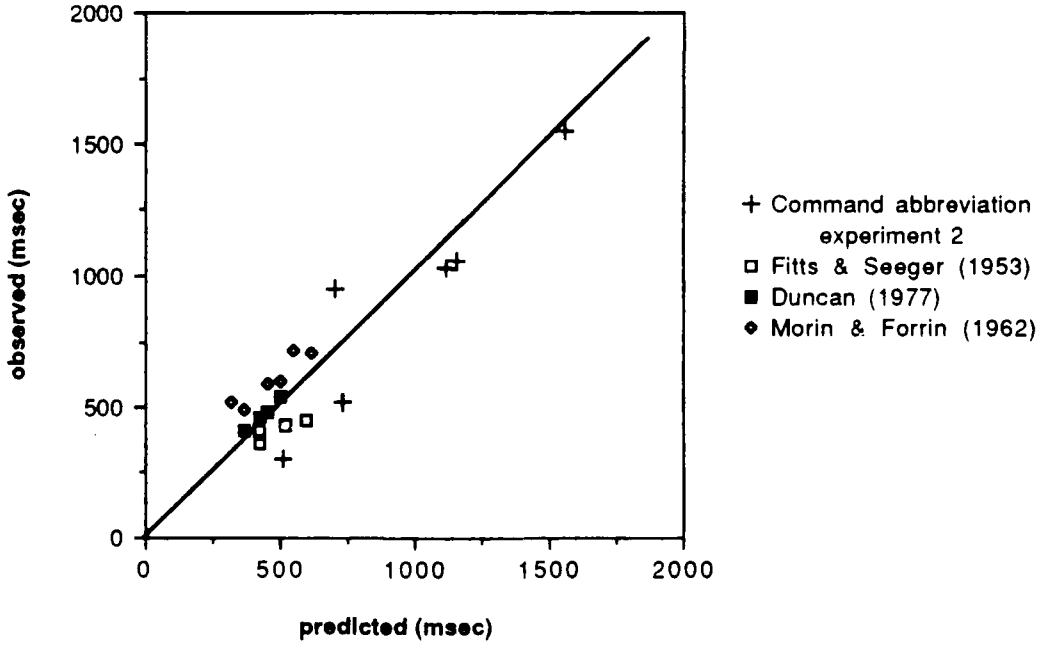


Figure 3-10: Predicted vs. observed response time for all four experiments.

retrieve the response given the stimulus). However, it can now be derived from the structure of the task and a sequence of cognitive operators.

Predictions of response time are dependent on a task analysis of the experimental, or real-world, situation. Some of the analyses appear simple, with only a few different algorithms (e.g., the Duncan

task). Others become quite complex requiring dozens of qualitatively different algorithms and scores of syntactically different Artless algorithms (e.g., the second abbreviation recall experiment). In either case, the task analysis is an important factor in making successful predictions. Yet, there are no rules or guidelines by which to construct good task analyses. This problem is not unique to GOMS or this thesis, but permeates cognitive modelling. In general, models of this sort are guesses at what mechanisms drive the cognitive processes involved in accomplishing a task. When a guess is good enough to explain, and even predict, performance, there is probably an element of truth in the model. If enough good guesses are accumulated, the elements of truth might be extracted to produce guidelines or rules for task analysis. Until that time, good guesses can serve as examples for future analyses. The analyses of the tasks in this chapter are such examples.

The operator durations estimated in this chapter are associated with specific operations of the MHP processors. Three of the operators used here are composed of several actions of the MHP processors and are operationally defined: (1) the perceptual operator perceives a written word and encodes it into an ordered list of letters that is the spelling, (2) the motor operator finds and hits a key on a keyboard (assuming an unskilled typist), (3) the retrieval operator retrieves an arbitrary, not well-learned association from LTM. In contrast, the cognitive operator is an elementary action of the cognitive processor. Whenever these operations are encountered in other tasks, their duration estimates (Table 3-9) can be invoked to produce predictions of task performance time. Testing this predictive power is the goal of the next chapter in this thesis.

Table 3-9: Perceptual, cognitive and motor parameter definitions and estimated durations.

Parameter	Definition	Duration
Perceptual Operator	Reading a word of about 6 letters and encoding it into an ordered list of letters	340 msec
Cognitive Operator	A cognitive processor cycle time	50 msec
Retrieval Operator	Recalling an arbitrary association that is not well-learned	1200 msec
Motor Operator	Typing a character on an alphanumeric keyboard at a rate of about 30 gwpm	230 msec

4. A GOMS Model of Expert Transcription Typing

Expert transcription typing was chosen as the next task domain into which to expand the use of GOMS and attempt to make quantitative, zero-parameter, a priori predictions of human performance. Typing is a major form of human-computer communication and is therefore an important skill to understand within HCI. It is also similar in many respects to the command abbreviation encoding task studies within SRC. It involves perceiving and internally representing words, manipulating the representations of those words, and the motor responses necessary to type characters. This similarity makes transcription typing only a small step away from the previously successful work.

Transcription typing provides a good domain of study, primarily because of a recent article by Timothy Salthouse (1986). Typing has been studied as a perceptual, cognitive and motor skill for over 70 years and there exists a substantial literature detailing the phenomena associated with typing, both frequent and robust phenomena and intriguing, but rare, phenomena. Salthouse distilled this literature and presented 29 robust phenomena associated with typing. He states his purpose:

> ...listing the major phenomena in need of explanation, and then providing an account of how proposed theoretical systems might handle these phenomena is an effective way of describing the properties of those systems...[the list of phenomena] defines the criteria by which alternative models in this area may be evaluated. (p. 304)

He goes on to state his criteria for selecting phenomena to include in the list, and his purpose in including them:

> ...first to identify phenomena for which convincing empirical support is available, and only then to venture possible explanations. In this manner one can at least be assured that the phenomena being explained by various theoretical models are genuine, and do in fact require explanation. (p. 304)

Thus, the "Salthouse 29" serves as an independent judgement of what phenomena require explanation by a model of typing. It would be better for engineering purposes if the Salthouse 29 were evaluated as to which of the phenomenon are important from an HCI design viewpoint, but, in the interest of objectivity and completeness, the Salthouse 29 will stand in lieu of less stringent engineering criteria.

This chapter focuses on expert typing. There are many indications that expert and novice behavior differs significantly. Experts exhibit skilled behavior where many processes have become routine. Novices may not know the position of all the keys on the keyboard and have to visually search for them; experts know where all the keys are and can hit them accurately without visual search. Gentner (1983) reports several qualitative and quantitative differences between the performance of novice typists and expert typists. His experimental work implicitly defines novices as typing at speeds up to 40 gwpm, experts as typing at speeds of 60 gwpm or better. Experts type faster than novices,of course; however, they also differ in their patterns of typing. Gentner examined the interkey intervals of four different classes of key transitions: 1-finger doubles where the same character is typed twice in succession, 1-finger non-doubles where two different characters are typed by the same finger, 2-finger sequences where two different characters are typed by two different fingers on the same hand, and 2-hand sequences where two different characters are typed by fingers on different hands. The pattern of interkey intervals for these different classes leads him to conclude that novice performance is limited by cognitive constraints (e.g., finding the key on the keyboard) whereas expert performance is limited by motoric or physical constraints (e.g., the distance between keys). Also, Gentner concluded that increased skill leads to less sequential, more parallel, performance. One implication of the novice/expert split is that the model of typing that was implicit in the SRC algorithms is not a model of expert typing. The model represented there was a sequential model that had each letter initiated by the cognitive processor and then executed by the motor processor, with no overlap of the processes. There is no reason to expect that this

sequential model, which fit the novice typists in the SRC research (averaging 31 gwpm in experiment 1 and 26 gwpm in experiment 2), would be appropriate for a model of expert typists, for, as Gentner concludes, "...with increasing skill the keystrokes in typewriting become less sequential; aspects of performance overlap in time to exploit the degrees of freedom inherent in the task." (p. 248). Expert typists demonstrate all of the Salthouse 29 phenomena, so restricting the range of speed does not shorten the list of phenomena in need of explanation. Therefore, the domain investigated was expert transcription typing, restricted to Gentner's definition of an expert, 60 gwpm and above.

Salthouse proposed a qualitative model of transcription typing, "derived from ideas introduced by earlier theorists (e.g. Cooper, 1983; Logan, 1983; Rumelhart & Norman, 1982; Shaffer, 1973, 1975a, 1976; Shaffer & Hardwick, 1970; Thomas & Jones, 1970)", to be used "as a heuristic device to help organize [his] review of the empirical literature". His model is based on four processing components: input, which converts text into chunks; parsing, which decomposes chunks into ordinal strings of characters; translation, which converts characters into movement specifications; and execution, which implements movements in a ballistic fashion. He provides qualitative explanations for the 29 phenomena within the framework of this model. As an indication of the state of theoretical development concerning typing, Salthouse says

> Ideally, this should be done for several alternative models simultaneously to provide a basis for comparative evaluation, but this is not yet practical, either because competing models have not yet been specified in sufficient detail to derive explanations or because the other models were intended to apply to only a limited set of typing processes. (p. 304)

For instance, a model by Rumelhart and Norman (Rumelhart & Norman, 1982) is "based upon an Activation-Trigger-Schema system in which a hierarchical structure of schemata directs the selection of the letters to be typed and, then, controls the hand and finger movements by a cooperative, relaxation algorithm." It is embodied in a computer simulation and provides detailed predictions about the movement of fingers, the response times for letters in different contexts, and several types of errors. However, Rumelhart and Norman explicitly state that their model does not cover

> ...the mechanisms used by inexperienced typists, nor the mechanisms involved in learning...the mechanisms involved in perception or the encoding of strings to be typed, nor in monitoring the accuracy of the typing...the deterioration of typing rate that occurs as the text is modified from normal prose to non-language or random latters...[or] non-alphabetical keys. (p. 6)

Sternberg, Knoll, and Wright (1978) present a subprogram-retrieval model. This model , again, is intended to apply to a subset of typing phenomena, namely the motoric aspects of typing. The model is based on research using a discontinuous typing task where a word, or string of characters, is presented well before the subject must type the string. Thus, "the perception of the material to be typed has presumably all occurred early in the trial, and the subject has plenty of time to rehearse or prepare in other ways for what he or she has to type" (p. 4). This model qualitatively accounts for several phenomena associated with discontinuous typing including the dependence of latency and interkeystroke interval on string-length, and a serial position effect on interkeystroke interval.

All of the existing typing models stand alone theoretically, rather than as part of a general theory of human behavior. Thus, a model based firmly upon the Model Human Processor, that attempts to explain most of the Salthouse 29, is unique in the field, because of its coverage, its powerful a priori quantitative predictions, and its theoretical base.

The proposed model of transcription typing is a direct application of GOMS to the domain. The basic

structure of the MHP, its three processors and their associated memories and principles of operation, is more evident in the GOMS model in this domain than in text-editing, because the behavior it attempts to explain is more immediate and closer to the architecture of the MHP.

4.1 Details of the GOMS Model of Expert Transcription Typing

The GOMS model can be applied to typing tasks performed by experts, given a few task-specific assumptions. The basic structure of the MHP, the goal structure of GOMS, plus these assumptions, form the GOMS Model of Expert Transcription Typing (METT).

4.1.1 Assumptions of the GOMS Model of Expert Transcription Typing

1. **BASIC ALGORITHM.** The basic algorithm is that a person perceives something (word, syllable, letter) using a perceptual operator. If it is a word or syllable, the spelling of that unit is obtained with a cognitive operator, the first character is initiated with a cognitive operator, and then executed with a motor operator. The second character is then initiated and executed, and so on. If a letter is perceived, then the character is initiated immediately following the perception, and executed. The Artless algorithm for the word or syllable process is as follows (the letter process skips L3).

		Operator Type
BEGIN	L 1	
Chunk ← Get-Chunk("To-Be-Typed-Copy")	L 2	Perceptual
Spelling ← Get-Spelling(Chunk)	L 3	Cognitive
Letter ← Initiate-Letter(First-Letter[Spelling])	L 4	Cognitive
Execute-Letter(Letter)	L 5	Motor
REPEAT BEGIN	L 6	
Letter ← Initiate-Letter(Next-Letter[Spelling])	L 7	Cognitive
Execute-Letter(Letter)	L 8	Motor
END	L 9	
UNTIL Done?(Spelling)	L10	
IF-SUCCEEDED Done?(Spelling)	L11	Cognitive
THEN END	L12	
END	L13	

2. **SERIAL/PARALLEL PROCESSING.** In the MHP, each processor works serially within itself, and concurrently with the other processors, with the following exceptions based on the typing task.
   a. . **PERCEPTION/COGNITION INTERACTION** Perception has to be complete before getting the spelling or initiation of a character can begin.
   b. **SAME-HAND CONSTRAINT.** A character on the same hand cannot be initiated with a cognitive operator until the motor processor execution of the previous character is complete.
   c. **PERCEPTION/WM LIMITATION INTERACTION.** The perceptual processor cannot perceive the next piece of information unless there is room in working memory for that information (see assumption 3 for the implication of this).
3. **WM LIMITATION.** In normal transcription typing, the perceptual processor stays three chunks ahead of the cognitive processor. The chunk is usually a word, but it can be a syllable or a character if words are not available in the specific typing task. This three-chunk limitation is the typical capacity of working memory proposed in the MHP. These chunks may access LTM for the spelling of one chunk at a time, extending the effective capacity of working memory to, on average, seven chunks (assuming an average word length of five letters, the seven chunks would be the five letters of the first word, plus the two next words).

4. **PERCEPTUAL CHUNKS**. The perceptual processor perceives at the most meaningful level available at or below the word level. For example, if words are present, they are perceived and encoded. If the view of whole words is restricted or if there are no words present (as when typing random letters), the perceptual processor perceives syllables. If syllables are not visible because of restricted view or random characters, then the perceptual processor perceives single characters.
5. **COGNITION/MOTOR INTERACTION**. Once a character is initiated with a cognitive operator, the motor operator that executes that character cannot be stopped.
6. **OPERATOR SIMILARITY**. Across all domains to which the MHP has been applied, similar operations involving similar perceptual and motor operators take similar amounts of time. Thus, the operator times estimated in the SRC experiments can be used here because the task in the SRC experiments was to perceive and encode a word, convert to its abbreviation, and type it, making the perceptual, cognitive, and motor operators similar to those in typing tasks.
    a. **PERCEPTUAL OPERATOR DURATION**. The time to perform a perceptual operator (perceive a visual stimulus) is 340 msec. A simplifying assumption is that this time is constant even if the thing to be perceived is a word, a syllable, or a character.
    b. **COGNITIVE OPERATOR DURATION**. The time to perform a cognitive operator is 50 msec.
    c. **MOTOR OPERATOR DURATION AND INTERACTION WITH SKILL**. As a simplifying assumption, practice in typing decreases the motor operator time only; the estimates of the perceptual and cognitive operators remain constant. There are undoubtably individual differences in the perceptual and cognitive processes (Salthouse, 1988), but given the amount of practice an adult typist has had perceiving words (as a part of reading) and in the cognitive operations involved in spelling words (as a part of writing), we assume that these operators remain constant relative to the more newly acquired motor operators of typing.

It is interesting to note the similarities between the METT and the the model proposed by Salthouse. His input component corresponds to the Get-Chunk perceptual operator, his parsing component corresponds to the Get-Spelling cognitive operator, his translation component corresponds to the Initiate-Letter cognitive operator, and his execution component corresponds to the Execute-Letter motor operator. The METT comes directly from the structure of the MHP; it's similarity to a model stemming from years of experience with the empirical data of typing lends credence to the MHP architecture. In addition, since the METT is a GOMS model within the MHP framework, it brings with it the parameter estimates associated with the MHP processor cycle times and therefore allows quantitative predictions to be made.

4.1.2 Estimating the Motor Operator

The METT says that the duration of the motor operator decreases with typing skill. In order to make predictions of different typing tasks with differently skilled typists, we need to provide estimates for the motor operator associated with different typing speeds. This derivation of the the motor operator will also serve as a an introductory example of how to use the METT.

Consider a typist typing a sentence from a standard typing test (the *example sentence*, which will be used in analyses throughout this thesis):

"One reason is quite obvious; you can get in and get out without waiting for the elevator."

The first three words are perceived with a perceptual operator, the spelling of the first word is retrieved from LTM with a cognitive operator, and the letters of the word, and the space following it, are initiated

and executed in turn. As soon as the space has been initiated, the chunk is out of working memory, making room for the next word, so the perceptual processor perceives the next word. The processes continue until the entire sentence is typed.

The parallel operation and sequential dependencies of the three processors make the processes of typing difficult to analyze and talk about. Fortunately, there is an analysis technique, borrowed from engineering project management that allows easy analysis of parallel resources (the three processors) working with sequential dependencies (outlined by the typing-specific assumptions). The technique is called critical path analysis. The particular version of this technique used for the analyses in this thesis is embodied in a project management software package for the Apple Macintosh family of computers, MacProject[7].

In a critical path analysis (Figure 4-1), each subtask necessary to accomplish a total task is represented as a box with a duration (e.g., "perceive a word" would be a subtask of transcription typing that would have a duration of 340 msec.). Dependencies between subtasks are represented by lines connecting the boxes (e.g., the cognitive subtask, "get the spelling" could not be started until "perceive a word" is completed, so a line would be drawn from "perceive a word" to "get the spelling"). The representation of a full task is called a *schedule chart*. The critical path is that set of subtasks that must be accomplished within their stated duration, or the entire task will take longer than anticipated.

Critical path was used to make estimates of the motor operator for typist of different speeds. For example, a 60 gwpm typist would be able to type the 89 characters of the example sentence in 17,800 msec (based on a five-letter average word). The schedule chart for typing the example sentence (Figure 4-1) is drawn up with the duration of all perceptual operators set to 340 msec, the duration of all cognitive operators set to 50 msec, and the duration of all motor operators set to an initial guess of, say, 200 msec (about the motor operator found in the SRC research). The critical path is calculated and the operators on the path are counted: 1 perceptual, 38 cognitive and 90 motor operators. Given the respective durations of these operators, the total time to complete this sentence would be

$$1 * 340 \text{ msec} + 38 * 50 \text{ msec} + 90 * 200 \text{ msec} \approx 20{,}240 \text{ msec.}$$

This is too long for a 60 gwpm typist, so the inital guess for the motor operator duration is too high. Iterating through this process yields a schedule chart with the same critical path, but the motor operator estimate giving the total time closest to 17,800 msec is 170 msec.

As well as getting numerical estimates from the critical path analyses, qualitative information about the roles of the three processors is obtained through inspection of the schedule chart. For instance, the perceptual operators are never on the critical path once the intial words have been perceived. This implies that the perceptual processes are not the limiting factors in the typing task. On the other hand, for a 60 gwpm typist, all the motor operators are on the critical path, indicating that a speed up of the motor operator will greatly affect the total time. At the other end of the speed spectrum, 160 gwpm, the critical path (Figure 4-2) looks quite different, with the cognitive operators determining the critical path and the motor operators playing a much less important role. This is because the duration of the motor operator is now shorter than that of the cognitive operator. This implies a theoretical maximum for typing, when the motor operator goes to zero, given the simplifying assumption that all the speed-up with skill comes from a decrease in the motor operator (Assumption 6c).

---

[7]*Apple, Macintosh and MacProject are trademarks of Apple Computer, Inc.*

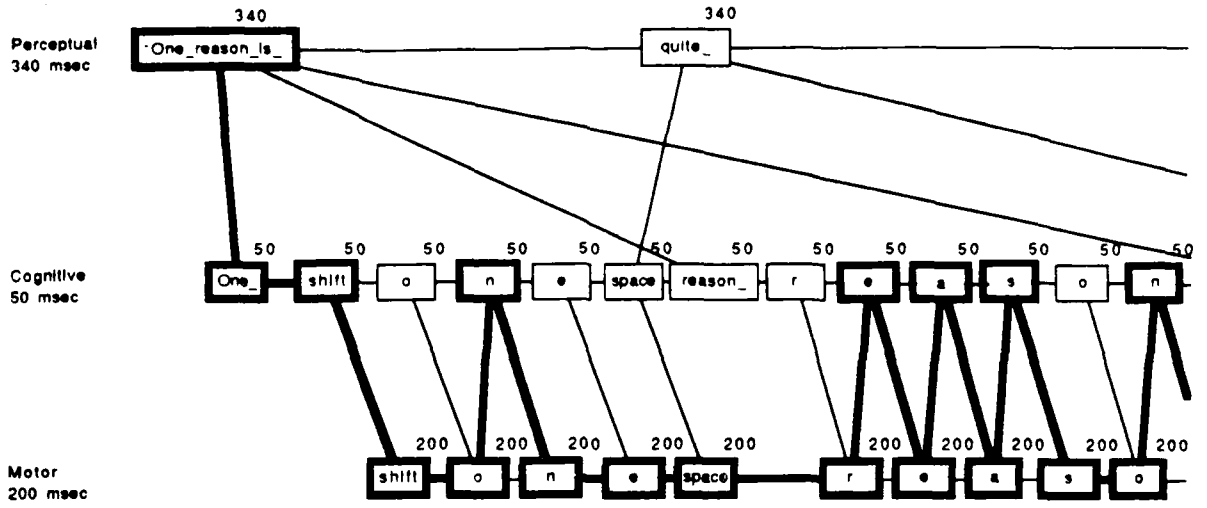


Figure 4-1: A schedule chart and critical path for the example sentence for a 60 gwpm typist with an initial motor operator estimate of 200 msec.

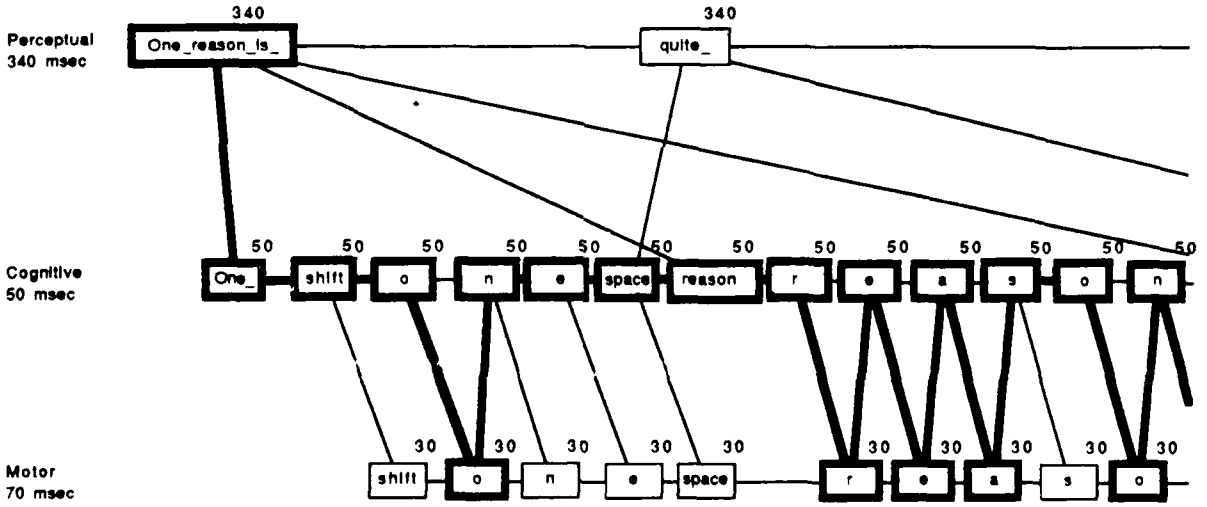


Figure 4-2: A schedule chart and critical path for the example sentence for a 160 gwpm typist.

Repeated application of this analysis process yields a chart of estimates of the motor operator vs. the gross speed of the typist (Figure 4-3). The motor operator estimates are rounded to the tens digit when used to analyze typing tasks.

Another useful analysis tool provided by MacProject is the conversion of a schedule chart into a *task timeline*. This chart (Figure 4-4) graphically depicts the time dependencies of the subtasks and was used in many analyses in this thesis. The critical path is indicated in the task timeline by boxes that have no grey area at their right side. The grey area is the *slack time* for each subtask, the time that the subtask could be delayed without affecting the duration of the total task. Subtasks on the critical path have no slack time, and thus no grey area.

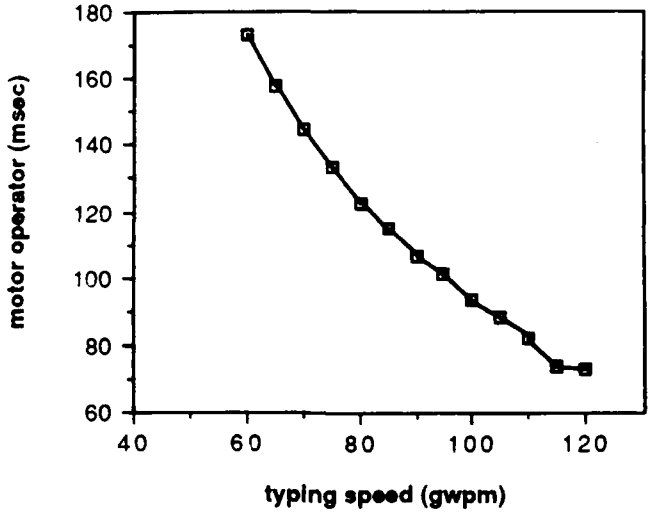


Figure 4-3: Typing speed vs. motor operator estimate.

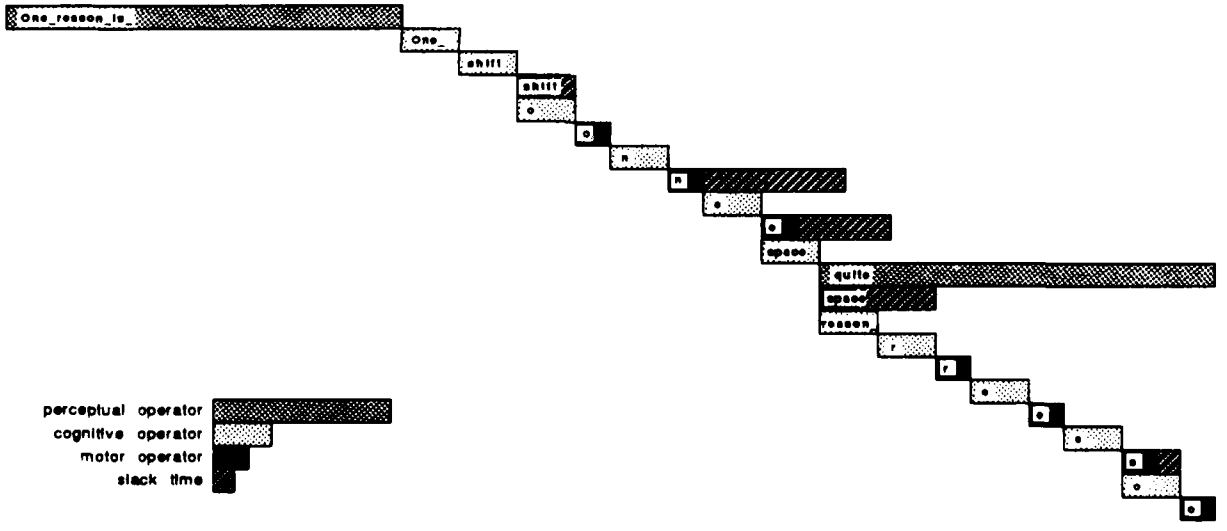


Figure 4-4: A task timeline representing the schedule chart in Figure 4-2.

Armed with the MHP, the typing-specific assumptions, the critical path analysis technique, and estimates for the motor operators of different speed typists (Figure 4-3), the 29 phenomena presented by Salthouse can now be analyzed in detail with the aim of making a priori, quantitative predictions.

4.2 Explaining the Salthouse 29 with the METT

Salthouse broke his 29 phenomena down into four categories: basic phenomena, units of typing, errors, and skill effects. This section follows the categorization and enumeration of the phenomena supplied by Salthouse, and uses them as a supply of phenomena and data for the METT to explain. Each phenomenon has been rated as to whether it is parametric (the results are given as a function of parameter, e.g., typing speed, preview window size), quantitative (an average numerical value is reported, but no significant pattern of results with another parameter), or qualitative (the phenomenon is a relative comparison rather than a numeric value). (Figure 4-5 displays these ratings.)

Corresponding to the ratings of the phenomena themselves, the METT account of each phenomenon is rated as parametric, quantitative, qualitative, or not covered. The METT makes response time predictions for any task that does not require distinguishing between fingers on a hand, as this is below the level of approximation chosen for the model. The METT also does not attempt to make predictions about the commission or detection of errors. Given these two constraints, the METT does not cover eight phenomenon (numbers 8, 18, 19, 20, 21, 22,and 23), although they will be discussed briefly with regard to the architecture of the MHP. As for the other phenomena, a METT prediction can be as precise as the available data in this hierarchy, but not more so. For example, predictions could be qualitative for a quantitative phenomenon, but not parametric. At the end, the success of the METT will be evaluated, compared to the available phenomena as characterized by Salthouse.

4.2.1 Basic Typing Phenomena

The first twelve phenomena are basic typing phenomena. These phenomena involve the speed of typing relative to other tasks (phenomena 1-3), how degradation of the text away from normal prose affects the rate of typing (phenomena 4-6), patterns of interkey intervals (phenomena 7-11), and the effects of a concurrent task (phenomenon 12).

4.2.1.1 Phenomenon 1: Typing is faster than choice reaction time

> People can type very quickly, with interkey intervals averaging only a fraction of the typical choice reaction time...[For an average typists of 63 wpm] the median interkey interval in normal transcription typing was 177 ms, whereas the median interkey interval for the same individuals in a serial two-alternative choice reaction time task was 560 ms.

From the serial/parallel assumption and the same-hand constraint assumption, the interkey interval during transcription typing is either one motor operator for digrams that alternate hands, or one cognitive operator plus one motor operator for digrams on the same hand. Thus, the average interkey interval for a 60 gwpm typist (motor operator = 170 msec), assuming an equal number of same- and alternate-hand sequences, would be 195 msec, 10.2% away from the 177 msec. observed for a 63 wpm typist.

Typical times quoted for a two-choice reaction time task are about 300-400 msec (Luce, 1986), which, although substantially lower than Salthouse's 560 msec, still supports the observed phenomenon. Salthouse' two-choice reaction time task was somewhat non-standard:

> Stimuli were uppercase and lowercase versions of the letters L and R, and responses were presses of the leftmost and rightmost keys on the lowest row on the keyboard, Z (for l and L) and / (for r and R). Subjects were instructed to respond as rapidly and as accurately as possible. Each keystroke caused the immediate display of the next stimulus until a total of 50 randomly arranged stimuli had been presented. (Salthouse, 1984a, p. 350)

Using the GOMS model of immediate behavior presented in the previous chapter, we can write a minimal algorithm for this particular two-choice reaction time task.

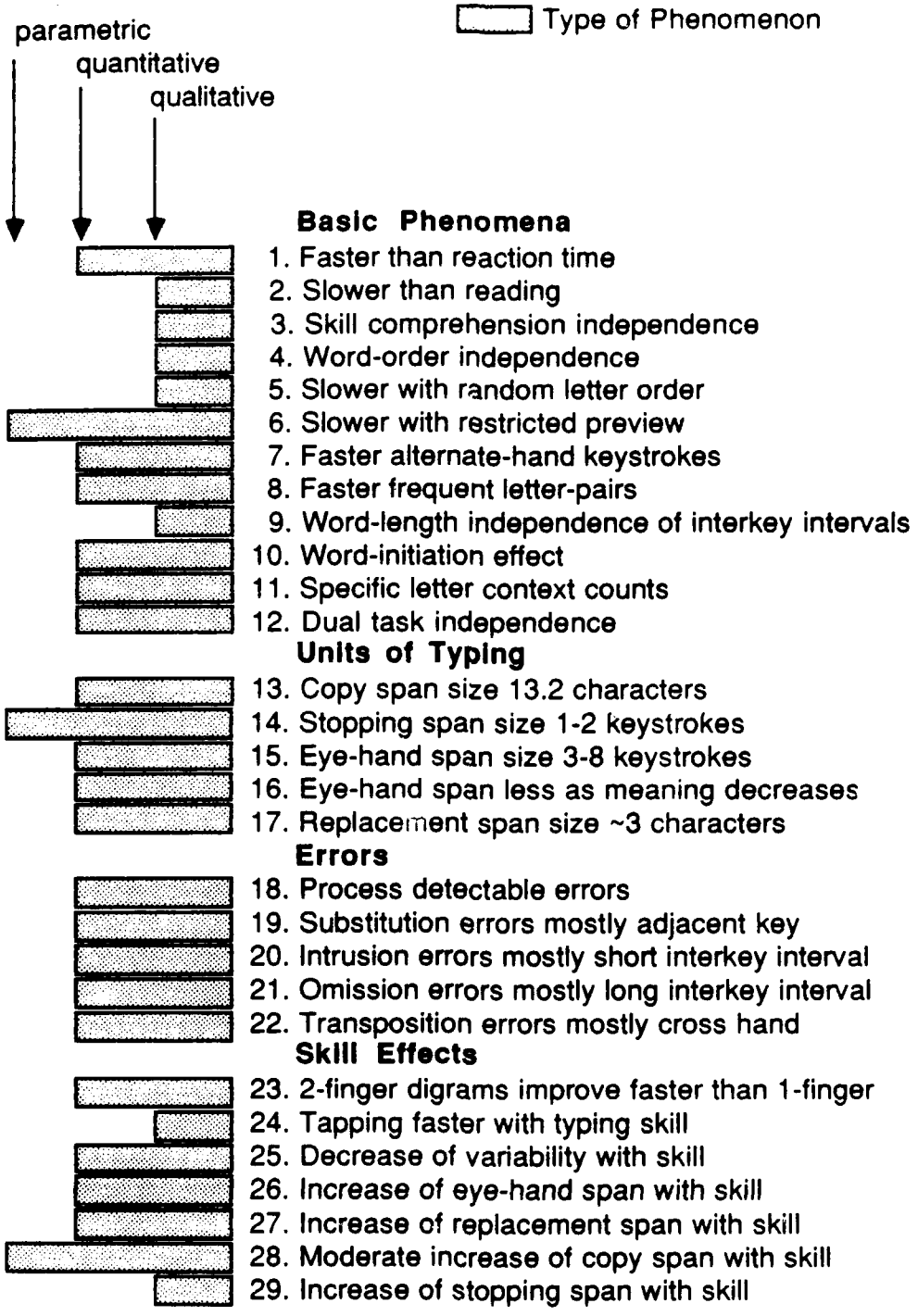


Figure 4-5: The Salthouse 29 and their ratings.

```
BEGIN                                          L 1
Letter ← Get-Stimulus("Letter")                L 2
IF-SUCCEEDED Is-R?(Letter)                     L 3
  THEN BEGIN                                   L 4
       Initiate(Right-Hand)                    L 5
       Execute(Right-Hand)                     L 6
       END                                     L 7
  ELSE IF-SUCCEEDED Is-L?(Letter)              L 8
         THEN BEGIN                            L 9
              Initiate(Left-Hand)              L10
              Execute(Left-Hand)               L11
              END                              L12
END                                            L13
```

In this algorithm, there are one perceptual operator (340 msec), two or three cognitive operators (for an average of 2.5 * 50 msec or 125 msec), and one motor operator (170 msec), for a total of 635 msec (13.4% above the observed 560 msec).

The METT explanation for this phenomenon is evident from the algorithms used to simulate the tasks. The typing algorithm allows overlap of the perceptual, cognitive and motor processes, and thus the average interkeystroke time (assuming equal same- and alternate-hand keystrokes) is the average of a single motor operator (interkey interval for alternate-hand keystrokes) and a motor operator plus a cognitive operator (interkey interval for same-hand keystrokes). In the CRT task, the algorithm is totally serial; the stimulus must be perceived, then cognitively processed, then the response is made. No overlap of processes means that all of the operators are on the critical path and contribute to the longer interkeystroke interval.

This is a quantitative phenomenon with a quantitative METT prediction.

4.2.1.2 Phenomenon 2: Typing is slower than reading

> Although typing is faster than reaction time, it is much slower than reading...two samples of typists averaged 246 and 259 words per minute when reading, but only 60 and 55 net words per minute, respectively, when typing.

The METT does not include a model of reading, so it cannot provide a predicted number corresponding to the 250 wpm reading rated quoted. However, we can interpret this assertion to mean that "input processes are generally not responsible for limiting the maximum rate of typing" (Salthouse, 1986). With a three-word look-ahead in normal transcription typing (Assumption 3), the perceptual processor is never on the critical path (except for the perception of the very first word). Therefore, the limit on the maximum rate of typing is not the perceptual process.

A sanity check is provided by the METT prediction of a theoretical maximum typing speed of 180 gwpm. This follows from the motor operator duration/skill interaction assumption, letting the motor operator time go to zero which provides an absolute upper bound. This maximum range is substantially less than the reading speed attained by many people (Salthouse's typists read at ~250 wpm on average), and, although there have been a few reported cases of world champion typists exceeding 180 gwpm, only 1% of typists ever go faster than 100 wpm!

This is a qualitative phenomenon with a qualitative METT prediction.

4.2.1.3 Phenomenon 3: Typing skill and comprehension are independent

Across typists, there is no relation between typing skill and degree of comprehension of material that has been typed.

There are no operators concerned with comprehending the perceived material in the algorithm of the METT. Comprehension is an entirely different process and, as such, would not be expected to correlate with typing speed in any way.

This is a qualitative phenomenon with a qualitative METT prediction.

4.2.1.4 Phenomenon 4: Typing rate is independent of word order

The rate of typing is nearly the same for random words as it is for meaningful text.

The METT works on the word level and has no comprehension or syntacticly high-level mechanisms, so random words will be treated no differently than meaningful text. Therefore, the METT predicts that the rate of typing would not be different for random words than it is for meaningful text.

This is a qualitative phenomenon with a qualitative METT prediction.

4.2.1.5 Phenomenon 5: Typing rate is slower with random letter order

The rate of typing is slowed as material approaches random sequences of letters.

The METT provides the following qualitative explanation for this phenomenon. Assumption 4 addresses the manner in which material is perceived and encoded in the METT. When the to-be-typed text is composed of words familiar to the typist, the words are stored away as chunks and accessed with the **Get-Spelling** cognitive operator. If the task is to type random letters, then there are no known words and they cannot be chunked as such. Therefore, the typist would have to drop down to the syllable level, where known syllables (e.g., pronounceable trigrams) could be chunked and then retrieved and unpacked. If the letters are so random that known syllables were not present, then the typist would have to drop down to the letter-level. Assuming a constant three-unit look-ahead, the perceptual processor will occasionally be on the critical path when very fast syllables are typed (e.g. alternate hand trigrams). At the letter-level, the perceptual processor will almost always be on the critical path. Thus, typing slows down.

A more detailed analysis of this phenomenon is based on the work of West and Sabban (1982). They used progressively degraded copy to test "the effects on performance of nonlanguage as compared to language materials". Examples of the five different test materials are shown in Figure 4-6. However, only the easy prose (EP), letter combinations (LC) and letter jumbles (LJ) conditions will be considered here because they address the phenomenon in question: the effects of random letters.

EP (Easy prose)	I have your letter in which you ask about the prices...
LC (Letter comb)	I veha uryo terlet ni chwhi ouy ska outab eth espric...
LJ (Letter jumble)	I evah uoyr rtleet ni hcihw oyu ska auobt teh rpcsei...
EW (Easy words)	Letter the I of about next ask in have month your which...
RW (Rare words)	Tycoon alp a si gumbo jamb boa em plop joist ouch piker...

Figure 4-6: West and Sabban's five types of materials.

West and Sabban presented their data broken down by speed of the typist performing the task; the

fastest group will be used as an example. Their average speed on a standard typing test was 100 wpm, corresponding to a motor operator of 90 msec.

A task analysis reveals three different strategies for typist performing this task. All of the strategies depend on a WM limitation of three chunks (Assumption 3). The first uses a 3-word look-ahead if words are available; if not, then a 3-syllable look-ahead if syllables are available, else a 3-letter look-ahead is used (hereafter this strategy will be called *3-w-s-l*). The second possible strategy is a 3-letter look-ahead only, where the typist decided to give up trying to pronounce the letter combinations, whether they are pronounceable syllables or not (hereafter, *3-l*). The last strategy is a 3-syllable look-ahead, if the whole word is made up of pronounceable syllables; if not, then the typist reverts to a 3-letter look-ahead (hereafter, *3-p-l*). The difference between 3-w-s-l and 3-p-l can be seen in the treatment of the word "rtleet". The former strategy would do "r", "t", "leet", and the latter strategy would do each letter separately because the whole "word" is not pronounceable. The EP material is composed of real words, so the only strategy needed is 3-w-s-l. The LC and LJ conditions might invoke the use of all three strategies. The effective typing speeds obtained with each of these strategies is shown in Figure 4-1.

Table 4-1: Effective typing speeds with each of the strategies.

	Effective Typing Speed (wpm)				
	Strategies			Straight	Weighted*
	3-w-s-l	3-p-l	3-l	average	average
EP	98	-	-	98	98
LC	99	84	45	76	65
LJ	71	57	45	58	51

*"weighted" means averaged across strategies 3-p-l and 3-l only for LC and LJ.

West and Sabban report their results as the percentage change between two conditions (shown in the OBSERVED column of Figure 4-2). In order to compare results, a weighting of these different strategies must be chosen. To make a true zero-parameter prediction, the weighting would have to be chosen from considerations other than the data. In the absence of external evidence for weightings, the minimum weighting assumption (equal weights across strategies) is used. The results of these predictions are shown in Figure 4-2.

Table 4-2: Observed and predicted percentage change between typing conditions.

	Observed	Predicted			
	% Change (sd)	Straight Average	[%err]	Weighted* Average	[%err]
EP vs. LJ	94.4 (29.4)	69.0	[26.9]	92.2	[2.3]
EP vs. LC	61.9 (22.1)	28.9	[53.3]	50.8	[17.9]
LC vs. LJ	20.5 (10.9)	31.0	[51.2]	27.5	[34.1]
Average absolute % error			43.8		18.1

*"weighted" means only averaged across strategies 3-s-p and 3-l for LC and LJ.

The equal-weighting assumption leads to predictions that are quite far from the observed differences between easy prose and the progressively more random letter orderings (average absolute percent error of 43.8%). If a slightly different weighting assumption is used, namely, that for the non-words, the typist uses only 3-p-l or 3-l, equally weighted, then the fit improves (average absolute percent error of 18.1%). Thus, the METT can predict the changes in typing speeds between the West & Sabban experimental conditions, but not without additional information about the mix of algorithms actually used by the participants.

This is a quantitative phenomenon with a qualitative METT prediction.

4.2.1.6 Phenomenon 6: Typing rate is slower with restricted preview

The rate of typing is severely impaired by restricted preview of the to-be-typed material.

This phenomenon has been reported by many researchers (Hershman & Hillix, 1965; Salthouse, 1984a; Salthouse, 1984b; Salthouse, 1985; Salthouse & Saults, 1987; Shaffer, 1973; Shaffer & French, 1971; Shaffer & Hardwick, 1970). A typical task description is found in Salthouse (1984a).

> Task 3 was to type material displayed in a single line of the video monitor and arranged such that each keystroke caused the display to move one space to the left. No visible copy was produced in this task. In successive conditions, the display contained 19, 11, 9 7 5 3 or 1 character of a 60- to 83-character sentence. The sentences were movie descriptions taken from TV GUIDE magazine and were randomly assigned to preview conditions.

The WM limitiation and perceptual chunks assumptions are particularly relevant for this task. As the preview is restricted, the three-word look-ahead is cut back to two-, then one-word look-ahead, then down to the syllable level and finally to the letter level. With a one character preview, the task is reduced to a series of choice-reaction time tasks with no look ahead.

If a whole word does not fit in the preview window, then the perceptual operator will not start until the word completed (or is shown to not to fit in the window). For instance, if the window is nine characters long, then the first view of the example sentence would be **One_reaso**. In this case, "One" would be perceived and encoded, but "reaso" would not be. The person would wait until until two characters were typed and the view became **e_reason_**; "reason" would now be a whole word and a perceptual operator could work on it. Note that the perceptual operator would not start until the punctuation after the word is visible (in this case, a space) because the "reason" in **ne_reason**, for example, could easily be part of "reasonable" or other longer word.

If a word is too long to fit into the window completely, then the syllables in the window are perceived and encoded. The remaining syllables in the word are perceived and encoded as they appear until the word is complete. Separating the words into syllables for this analysis can be done from two different viewpoints: the subject's viewpoint and the viewpoint of the text. For example, with a five character window, the phrase "the elevator" would not fit; after typing the "the", the view would be **_elev**. The typist would not know what word was coming and might choose to encode this as either "e" and "lev", "el" and "ev", or "el" and "e" waiting for more letters before encoding the "v". On the other hand, the text is actually "elevator" and this is commonly broken up as "el", "e", "va", "tor" (Webster's New Collegiate Dictionary, 1979). METT analyses will always be made from the viewpoint of the text, making them simple, straightforwward and objective. Thus, **_elev** would be encoded as two syllables: "el" and "e". With a three-syllable look-ahead, this sequence would not fill up working memory Therefore, as soon as the space is typed, the view would be **eleva** and the perceptual processor would perceive and encode the

next syllable, "va". After the "l" is initiated, there would be room in working memory, but the view would be **levat** and no new syllable would be visible. After the next "e" is typed, however, **vator** becomes visible and the "tor" is perceived and encoded. Then, after the "v", the view becomes **ator.** and the "." is perceived and encoded separately. Figure 4-7 shows the schedule chart depicting these dependencies.

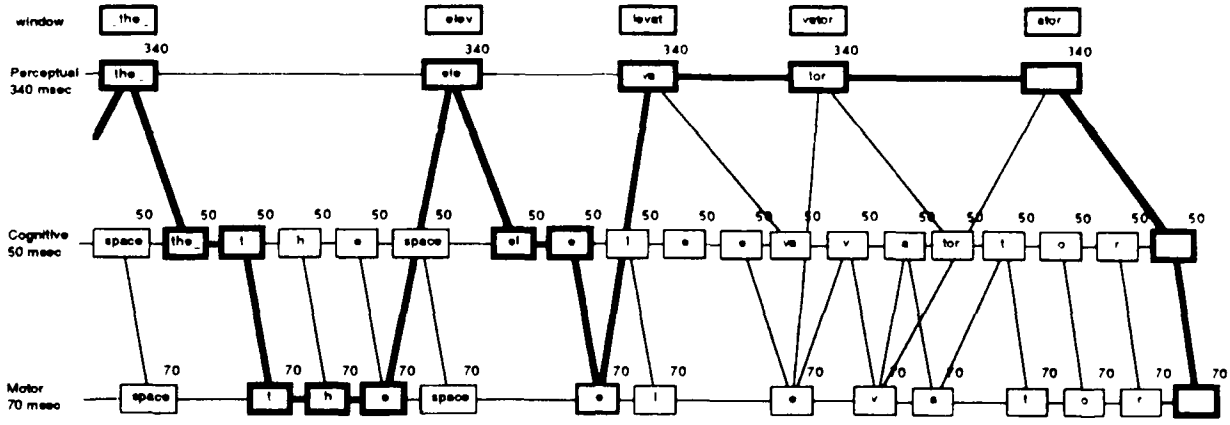


Figure 4-7: Schedule chart for "the elevator" in a five-character preview window.

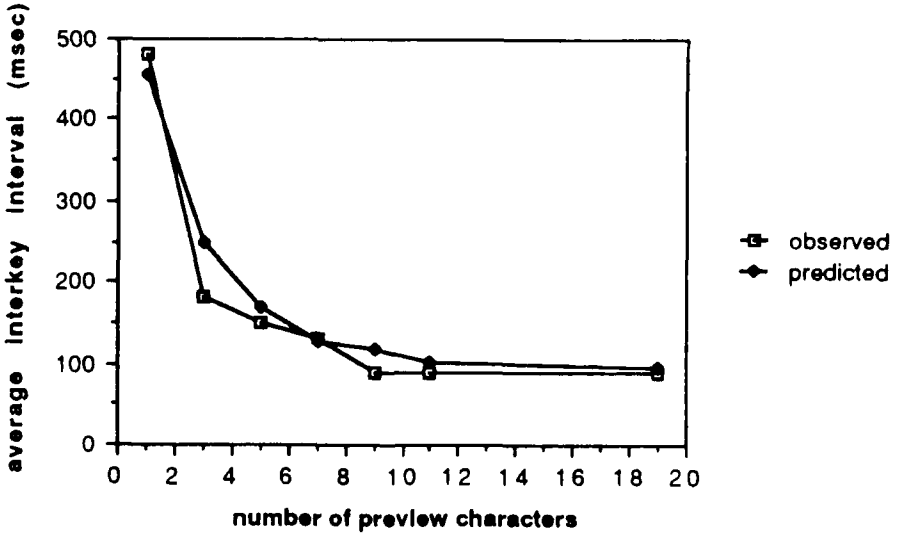


Figure 4-8: Restricted preview predicted and observed results.

An analysis was done using the example sentence, all of Salthouse's conditions and a 120 gwpm typist (Salthouse reports the data for his fastest participant, a 117 gwpm typist). The results of the analysis appear in Figure 4-8. The average absolute percent error is 15.8%.

This is a parametric phenomenon with a parametric METT prediction.

4.2.1.7 Phenomenon 7: Alternate-hand keystrokes are faster than same-hand keystrokes

Successive keystrokes from fingers on alternate hands are faster than successive keystrokes from fingers on the same hand.

The serial/parallel processing assumption with the same-hand constraint (Assumptions 2 and 2b) imply this result. Because a keystroke cannot be initiated by the cognitive processor for the same hand until after the current keystroke is completed by the motor processor, same-hand keystrokes have a cognitive operator on their critical path that does not appear in the alternate-hand keystrokes. (See the difference between the o-n sequence and the n-e sequence in Figure 4-1, page 50.) Thus, the METT predicts that the interval between same-hand keystrokes will be 50 msec longer than the interval between alternate-hand keystrokes. Salthouse reports a range of 30 msec to 60 msec; 50 msec is 11.1% above the mean of this range.

This is a quantitative phenomenon with a quantitative METT prediction.

4.2.1.8 Phenomenon 8: More Frequent Letter Pairs are Typed More Quickly

Letter pairs that occur more frequently in normal language are typed faster than less frequent pairs.

The METT does not try to explain this phenomenon, because the model simplifies the situation by differentiating only between hands, not between fingers. However, only 4% of the variability in interkeystroke interval is accounted for by the log of the frequency of occurrance, when the effect of same-hand vs. alternate-hand sequences is eliminated, so this is a relatively small effect when compared to the many larger effects in this list (Salthouse, 1984b).

This is a quantitative phenomenon, although a small one, that is not covered by the METT.

4.2.1.9 Phenomenon 9: Interkey Intervals are Independent of word-length

There is no systematic effect of word length on either the interkey interval between the space and the first letter in the word, or on the interkey interval between letters within the word.

The METT works at the word level and the operator that retrieves the spelling of a word takes the same amount of time no matter what the length of the word. Therefore, the METT predicts no word-length effects.

This is a qualitative phenomenon with a qualitative METT prediction.

There is a word-length effect in discontinuous typing tasks (where a word is displayed only after the typing of the previous word is complete) (Sternberg, Knoll, & Wright, 1978; Larochelle, 1983). For the METT, this implies that the word-length effect is in the perceptual processor, not in the cognitive processor. Several researchers have found that words with more letters take longer to read (Henderson, 1982) and this may be the source of the word length effect. These reading results could easily be incorporated into a GOMS model of discontinuous typing, but it is not necessary to introduce this complexity into the model of transcription typing because the perceptual processing is not on the critical path and thus the effect is not evident.

4.2.1.10 Phenomenon 10: The first keystroke in a word is slower than subsequent keystrokes

The first keystroke in a word in normal continuous typing is generally slower than subsequent keystrokes in the word.

The METT predicts that the first character of a word would have a longer interkey interval than subsequent letters whenever the cognitive operator that gets the spelling of the word is on the critical

path. This situation occurs for all typists with speeds greater than 90 wpm. However, the average increase is very small because only one same-hand and alternate-hand character sequence brings the get-spelling operator to the critical path (i.e., the last letter of the previous word is typed by the right hand, and the first letter of the word in question is types by the left hand). Thus, the increase in interkey interval for the first letter of a word ranges from 0% for a 60 wpm typist to 8.4% for a 120 wpm typist, for an average of 2.3%. Salthouse (1984a, 1984b) reports a 20% increase overall with a 60 wpm average typist Thus, the METT predicts a word-initiation effect, but an order of magnitude less than the observed performance.

This is a quantitative phenomenon with a qualitative METT prediction.

The METT makes an interesting second order prediction, that the word-initiation effect increases at the high end of the expert range. This is counter to the results reported by Salthouse (1984a). He obtained about a -0.4 correlation between skill and word-initiation effect. However, his typists spanned a much larger range of skill than the METT addresses. It is possible that a mechanism different from accessing the spelling of the word exists in novice typing, causing a large word-initiation effect at the lower speed levels (e.g., perhaps a novice perceives a word, then types it, then goes back to the copy to perceive the next word). It is possible that the word initiation effect all but disappears in the low expert range, when the typist begins to look ahead of the word currently being typed, and then reappears slightly at the highest skill levels because of the word-access mechanism in the METT. Such a pattern could produce the correlation found by Salthouse. The data for different skill levels is not reported in the literature, so this pattern cannot be verified at this time. However, it is an empirically testable prediction, on the agenda for further investigation.

4.2.1.11 Phenomenon 11: The time for a keystroke is dependent on the specific context

The time needed to produce a keystroke is dependent on the specific context in which the character appears.

The "specific context" quoted above means the specific characters surrounding the character being typed, both ahead of and behind that character. Previously explained phenomena partially account for this effect. Phenomenon 7 (the same-hand/alternate-hand difference) phenomenon 8 (the digram frequency effect) and phenomenon 10 (the word initiation effect) all contribute substantially to this phenomenon. The METT provides explanations for phenomenon 7 and 10. However, after these effects have been controlled for in careful experimentation, a context effect remains and is active over as many as three characters of context. Models that include the detailed behavior of each finger (e.g., Rumelhart & Norman, 1982) can predict these effects, but the coarse grain of the METT make this effect beyond the scope of the model. However, the locus of the explanation is probably within the motor processor and extension of the model with regards to more detailed motor programs could provide an explanation.

Consider the following extension of the METT to explain data presented by Rumelhart and Norman (Rumelhart & Norman, 1982). These researchers used a detailed computer model to simulate the interkey interval between the 66 most common digraphs in a large corpus of typing data. Overall, their simulation accounts for 74% of the variance. They present actual performance data for only nine digraphs, those involving the index and middle fingers of the left hand, always ending with the striking of the *e* key. Their simulation produces estimates of interkeystroke times for these nine digraphs in "arbitrary model units", which, when I regressed them against the observed times, account for 69% of the variance ($p < 0.01$).

According to the principles of operation of the MHP (Section 2.2), movement time follows Fitts Law, so, if an extension to the METT can explain this fine structure of interkey movement, it will do so with a Fitts Law analysis. To do this analysis, keyboard with a 0.5 in. square key, and a 0.75 in. key separation, center-to-center, was assumed (Figure 4-9). For same-finger digraphs, it was assumed that the finger had to move the whole way, center-to-center. For different-finger digraphs, it was assumed that the middle and index fingers stayed in the same relative horizontal position as the index finger moves to strike a key, and that striking a key on the upper or lower row moves the middle finger only half way to that row, along the trajectory it would take if that relationship were maintained. Following this assumption, the positions of the middle finger when the index finger hits another key are marked on Figure 4-9 with the lower-case letter of the key being hit. The straight-line distance that the middle finger had to travel was called the "Distance Moved".

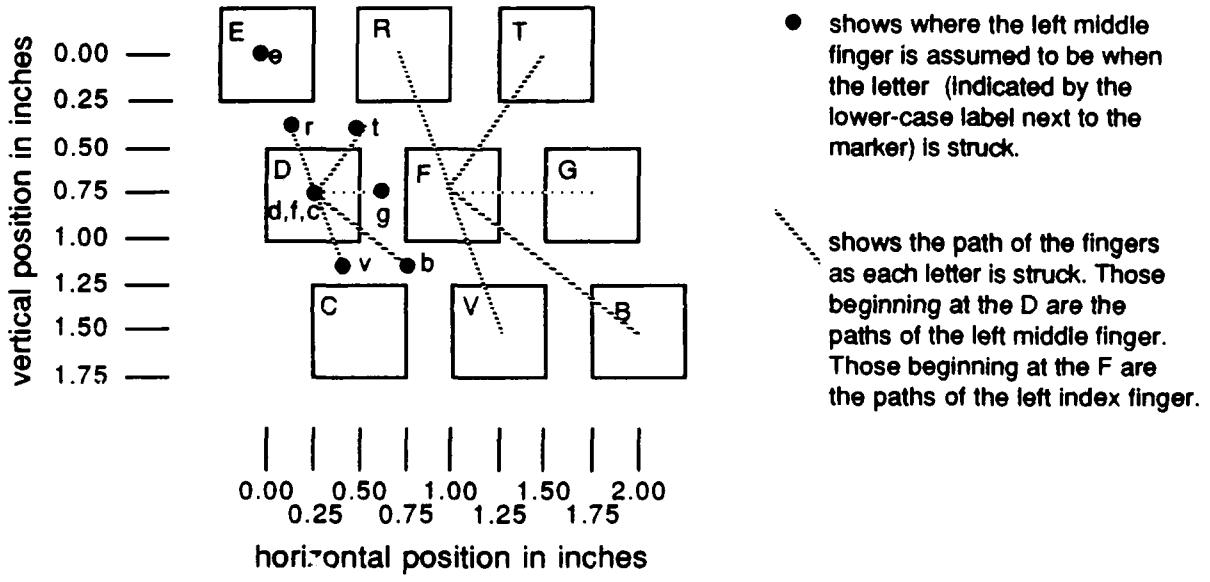


Figure 4-9: Keyboard and position of the left middle finger when each key is hit.

It was assumed that for the same-finger transitions, the finger was lifted up, then moved over the *e* key, then pressed down. Since the *e-e* transition has no movement time, only up and down, the up and down movements were defined by the observed time to make the *e-e* transition; each were assumed to take 83 msec apiece. These ups and downs were subtracted from the same-finger times to get the pure movement time.

To get the pure movement time for the different-finger transitions, it was assumed that pressing the *f* key leaves the middle finger on the *d* key so the *f-e* transition involves the same pure movement time as the *d-e* transition. From that assumption, the down time (83 msec) and the pure movement time for the *d-e* transition (35 msec, which is the total observed time minus the up and down time) was subtracted from the 168 msec observed for the *f-e* transition, to get an estimate of how much time is spent picking the left index finger up before starting to move the middle finger: 50 msec. This value is less than a pure up time (83 msec), indicating, as Rumelhart and Norman observe, movements of different fingers can overlap. This estimate plus the down time (133 msec) was subtracted from all the different finger observed times to get the pure movement times.

Predictions for the pure movement times were made by using the Fitts' Law:

$$T_{pos} = I_M \log_2(2D/S),$$

where I_M is a free parameter established by regression. These predictions were regressed against the estimated pure movement times, forcing the regression through zero, to get an estimate of I_M: 20.70 msec.

The total predicted time is then calculated by

$$T_{pred} = 20.70 * \log_2(2D/S) + \text{Up-time} + \text{Down-time}$$

where D is the Distance Moved, S is the size of the key, 0.5 inches, Up-time is 83 msec for same-finger movements and 50 msec for different-finger movements, and Down-time is 83 msec.

The Rumelhart and Norman total time prediction was gotten from regressing the simulation's results against the observed times and finding the coefficient of the slope (9.17449) and intercept (104.52045) of the best-fit line and then using the equation

T(pred) = 104.52045 + 9.17449*simulation-result.

The observed times and the two different predicted times can be found in Table 4-3. The percent error for each key transition was calculated for both the predictions and the average absolute percent error for each prediction technique also appear. The MHP predictions based on Fitts' Law, and containing only one free parameter (I_M), are slightly better than the Rumelhart and Norman predictions, containing two free parameters (slope and intercept).

Table 4-3: Comparison of time predictions made by the METT and Rumelhart & Norman's simulation.

		----------MHP (Fitts' Law)----------				-----Rumelhart & Norman------		
Keys	Tot Time Obs (msec)	Dist. Moved (inches)	Est. Move Time (msec)	Total Time Pred (msec)	%err	Simulation Units	Tot Time Pred (msec)	%err
e-e	165	0.00	0	166	-0.6	5.9	159	3.6
d-e	201	0.79	34	195	3.0	9.5	192	4.5
c-e	215	1.58	55	221	-2.8	12.7	221	- 2.8
f-e	145	0.40	14	146	0.7	7.1	170	-17.2
t-e	159	0.63	28	157	1.3	7.0	169	- 6.3
f-e	168	0.79	35	162	3.6	7.3	171	- 1.8
g-e	178	0.98	41	174	2.2	7.2	171	3.9
v-e	178	1.19	47	180	-1.1	7.5	173	2.8
b-e	195	1.35	50	183	6.2	8.1	179	8.2
		Average absolute percent error =			2.4	Average absolute percent error		= 5.7

When regressed against the observed interkeystroke times, these METT estimates account for 99% of the variance ($p < 0.01$). Using the data to estimate the slope of the Fitts' Law equation, produces predictions with an average absolute percent error of 2.3%.

This example of a possible extension to the METT explains the single-character context effect quite well. It does not make zero-parameter predictions, in that the slope of the Fitts' law curve was determined by regression against the data, so the predictions are qualitative, not quantitative. If the METT were to be

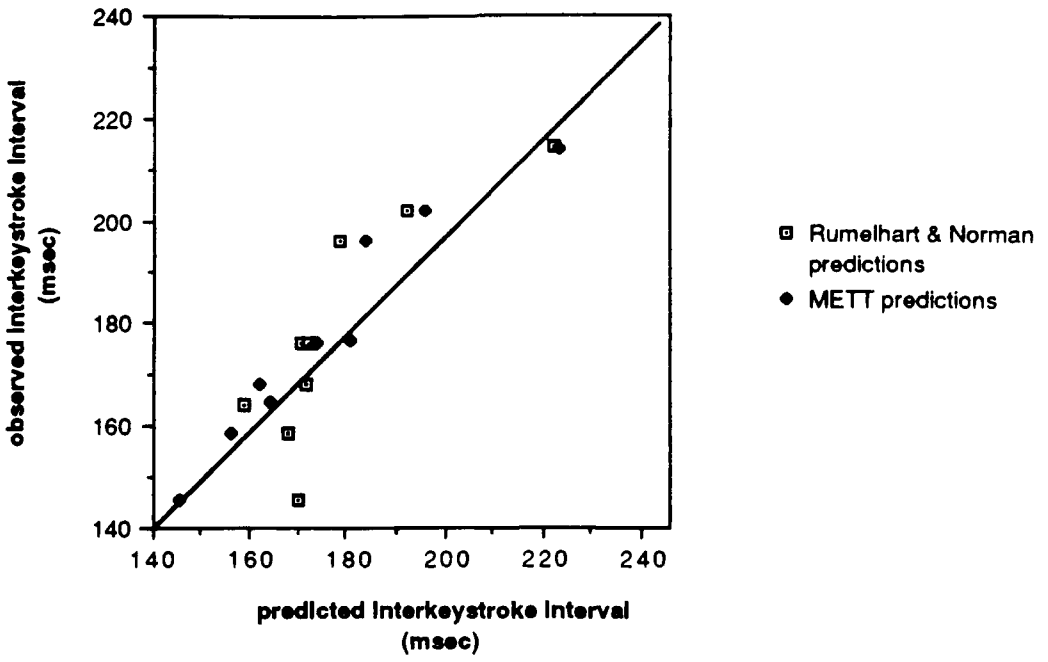


Figure 4-10: Predictions of movement time made by Rumelhart & Norman's model and by the METT.

extended to this level of detail, the next step would be to obtain performance data for other digrams and make true zero-parameter predictions.

With the METT as presented in Section 4.1.1, without the extension, this is a quantitative phenomenon that is not covered by the METT. With the extension, qualitative predictions can currently be made.

4.2.1.12 Phenomenon 12: A concurrent task does not affect typing

With highly skilled typists, a concurrent activity can often be performed with little or no effect on speed or accuracy of typing.

Salthouse and Saults (1987) gave typists a simple reaction-time task and a task where they had to type and perform the reaction-time task concurrently.

> The second task was an auditory reaction-time task in which responses were made by pressing a foot pedal containing a micro-switch.
>
> ...Task 3 was [where]...subjects typed from printed copy while responding to auditory signals with foot pedal responses. The typing task was stressed both by instructions and by delaying the introduction of the concurrent reaction-time task until subjects had typed for about 30 s.

The results of these tasks are presented in Table 4-4. The typing task was not slowed by the concurrent reaction-time task, but the reaction-time task slowed down considerably, indicating that the reaction time task had not yet become automatic and that the devices used to emphasize the typing task were successful.

The first step in a GOMS analysis of this concurrent task situation is to estimate a new motor operator parameter for pressing a foot pedal by writing an algorithm for the simple reaction-time task and fitting the motor parameter to the observed performance. An Artless algorithm for the simple reaction-time task is as follows.

Table 4-4: Results of typing and reaction-time tasks, alone and concurrent.

	alone	concurrent
typing interkey interval (sd), in msec.	181 (64)	185 (62)
reaction time (sd), in msec.	269 (49)	431 (85)

```
BEGIN
Tone ← Get-Stimulus("Tone")                          100 msec
IF-SUCCEEDED Right-Tone?(Tone)                        50 msec
 THEN BEGIN
        Initiate-foot-press                           50 msec
        Execute-foot-press                            new motor operator
        END
END
```

Subtracting the perceptual and cognitive operator times from the observed reaction-time, 269 msec, leaves an estimate of the motor operator to press a foot pedal of about 70 msec. This value happens to be the typical motor processor cycle time given by Card, Moran and Newell.

With this new motor operator estimate, the algorithm for the simple reaction-time task is superimposed on top of the METT algorithm for a 60 gwpm typist typing the example sentence (Salthouse and Saults' typists averaged 63.1 nwpm). The tone was assumed to start at 25 random locations within the example sentence and each location was analyzed in isolation, to simulate the random positioning of the tone within a longer portion of text (Salthouse and Saults used up to 250 words). Since the reaction time task was not automatic and the typing task was emphasize, the operators that perform the reaction time task were woven in between those of the typing task, with the typing operators taking precedence. If the perceptual processor was not busy perceiving a word when the tone started, then the perception of the tone began at the onset of the tone, otherwise the perception of the tone began as soon as the perceptual process completed the perception of the word. When the perception of the tone was complete, if the cognitive operator was not busy doing something for the typing task, the verification of the tone began, otherwise the verification waited until the typing cognitive operation was complete. Then the cognitive operators for the typing task and the reaction-time task were woven together, alternating between tasks if they were competing for cognitive processing time.[8] The motor operator to press the foot pedal began after the foot-press was initiated by the cognitive processor and the motor processor was not busy typing a character. (See Figure 4-11.)

The concurrent reaction time was predicted from these schedule charts by measuring the time between the (randomly defined) onset of the tone and the completion of the foot-press motor operator. The average reaction time predicted by this analysis is 435 msec, 0.9% away from the 431 msec observed by Salthouse and Saults.

---

[8]Sometimes the typing task involved a same-hand key sequence and the cognitive processor was waiting for the completion of the motor processor. This left enough time for the reaction-time cognitive operators to execute without alternation between tasks.

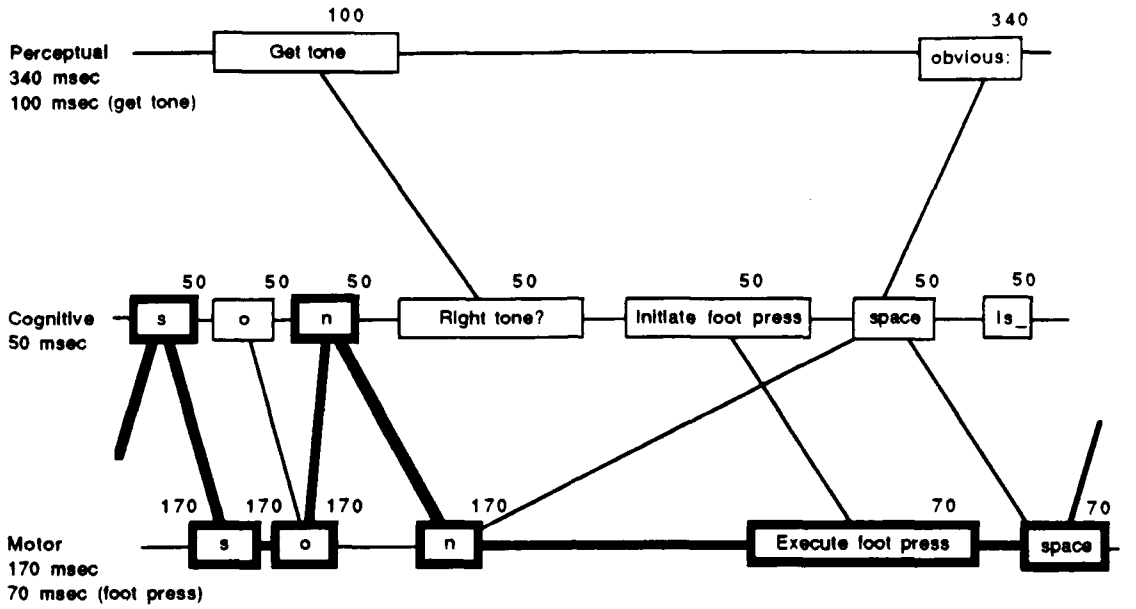


Figure 4-11: Schedule chart for the concurrent typing and reaction-time tasks.

The effect on the average interkey interval for Salthouse and Sault's task is reported to be small. From Table 4-4, the mean of the interkey interval was 181 msec for the normal typing task and 185 msec when the concurrent reaction time task was added. The METT predicted an average interkey interval of 195 msec for the normal typing task, 7.7% above the observed value. The interkeystroke interval for those keystrokes that were interrupted by the reaction-time task was predicted to be 240 msec. However, this was only for those keystrokes that were directly involved in the concurrent task, not the average for the entire typing task. Salthouse and Saults state that the task is a combination of Tasks 1 and 2 in the same study, which they describe in detail. If the combination of tasks is taken literally, then there were 30 tones presented within a 1200 character passage. (This ratio of tones to characters was indeed the case, Salthouse, private communication, 1988.) Thus, 30 interkeystroke intervals increased to 240 msec and 1170 keystrokes remained at an average of 195 msec. With this ratio, the overall average interkey interval for typing with the concurrent task was predicted to be 196 msec, 6.0% above the observed time. This analysis supports the claim that a concurrent task has little or no effect on the typing speed of an expert typist. An interesting prediction of the METT is that the interkeystroke intervals occurring as the foot-press is occurring will increase; this prediction is left for future empirical verification.

This is a quantitative phenomenon with a quantitative METT prediction.

4.2.2 Units of Typing

The next five phenomena deal with various units of typing: how many characters can be typed after the copy has been removed without warning (copy span), how many characters are typed after the typist has been told to stop (stopping span), how many characters ahead of the fingers must the typist's eyes be when typing at assymptotic speed (eye-hand span), and how many characters ahead of the fingers can a change be made in the copy and still be reflected in the performance (replacement span).

4.2.2.1 Phenomenon 13: Copy span is 7-40 characters

> The copying span, defined as the amount of material that can be typed accurately after a single inspection of the copy, ranges from two to eight words, or 7-40 characters.

This has been measured in many different ways, and the different methods yield vastly different results, accounting for the wide range. For instance, Rothkopf (Rothkopf, 1980) measured the copy span by asking the typists to glance at the copy, remembering as much as possible, and then type it out before glancing at the copy again. This is a very different task than normal transcription typing and it yielded the result that a typist can remember up to 40 characters at a time. Salthouse (Salthouse, 1985) measured the copy span in a way more appropriate to transcription typing, and got an average copy span of 13.2 characters for all typists (speed range from 20 to 120 gwpm), but an average of 14.6 characters for expert typists (above 60 gwpm).

The Salthouse experimental situation was as follows.

> The procedure involved presenting material on the video monitor using the left-ward moving display with a preview window fixed at 39 characters. After a predetermined number of keystrokes, the display was erased and the typist instructed to continue typing as much material as he or she was confident appeared on the display. The material consisted of eight sentences, movie descriptions from TV GUIDE magazine, with an average length of 75 characters. Two sentences each were typed with 15, 25, 35 and 45 keystrokes prior to the disappearance of the display. The median number of characters that were typed correctly after the blanking of the display served as the measure of copying span.

The METT can easily model this task. Predictions can be made at several different levels of detail. First, a quick and dirty analysis is presented, then a more detailed analysis of the Salthouse task.

On average, a word is 5 letters long. If there is a 3-word look ahead, there is, on average, 15 letters in the perceptual buffer, or working memory. If the display is removed randomly, it will be removed, on average, 2.5 characters into a word. Therefore, there will be a copy span of about 2.5 words, or 12.5 characters. This prediction is within 14.4% of the observed 14.6 characters.

To do the more detailed analysis, more detailed information is required:

> Net typing speeds for the 29 typists ranged from 18 tp 113 NWPM with a mean of 62.4. Gross speeds ranged from 20 to 120 words per minute with error percentages from 0.1 to 4.2. ...four speed groupings (six typists at less than 40 NWPM, M=27.8; seven typists at between 40 and 60 NWPM, M=48.1; eight typists at between 60 and 80 NWPM, M=72.5; and eight typists with speeds greater than 80 NWPM, M=90.6).
>
> The correlation between NWPM and the median number of characters typed after the disappearance of the test display (i.e., the copying span) was .35 (.10>p>.05). Across all typists the copying span averaged 13.2 characters, and from the slowest to the fastest speed groups the spans averaged 10.5, 12.4, 15.5, and 13.6 characters, respectively, $F(3,25) = 1.75$, $p>.15$.

Salthouse used different sentences and four different stopping points, but the same effect can be gotten by using only the example sentence imposing a stop after every character, and averaging them all together. Assuming an 80 wpm typist (the approximate average of the typists in the expert range, above 60 wpm), the copying span was predicted as if the display disappeared at each position between character 1 ("O" in "One") and character 75 ("r" in "for"); after character 75 there are no more characters to see with look-ahead. It is assumed that there is a three-word look-ahead (WM limitation and perceptual chunks assumptions) and as soon as the last character of a word (including the punctuation and space after it) is out of working memory, i.e., the cognitive processor has sent a signal to the motor processor with an initiate-letter operator, the next word can be perceived.

The disappearance of the copy is triggered by the typing of a character, which is the completion of a motor operator. Since the initation of characters by the cognitive operator triggers the perception of the next word, the perception of the word takes a finite amount of time, and the cognitive and perceptual processors work in parallel with the motor processor, the relationship between the character just typed and the contents of working memory is not a straightforward one. The relationship is determined by the same- and alternate-handed history of the text being typed and the duration of the operators. A task timeline for the sentence being typed is necessary to chart the contents of working memory at every stopping point. The copy span is how many letters are typed after the copy disappears, so it is a combination of the letters that have already been initiated by the cognitive processor (but not yet executed by the motor processor) and what is left in working memory, which can be initiated and executed.

Figure 4-12 shows a small portion of a task timeline from which the copy span can be determined. Table 4-5 shows the letters that can be typed for several different stopping points. The copy span for the average typist in the expert range (~80 gwpm), from this detailed calculation is 12.1 characters (16.9% from the observed 14.6 characters).

This is a quantitative phenomenon with a quantitative METT prediction.

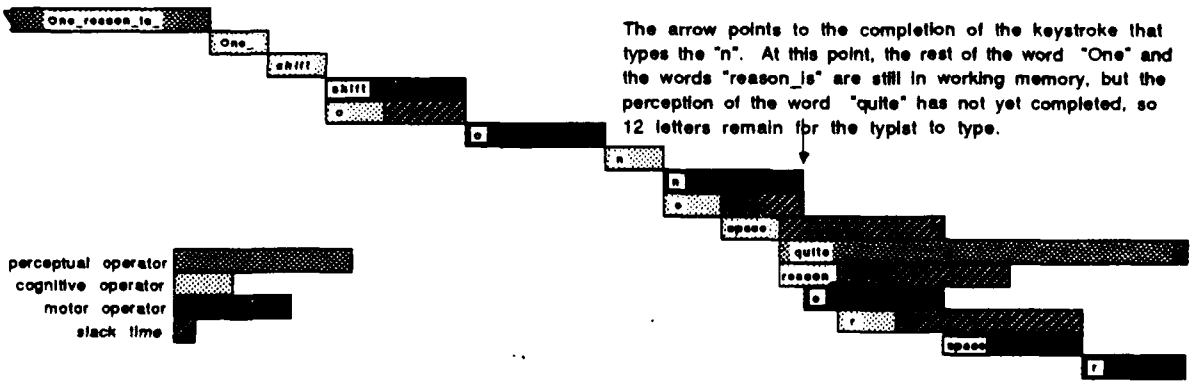


Figure 4-12: Portion of a task timeline for an 80 gwpm typist showing the copy span analysis.

The first, quick and dirty analysis gives a good prediction of the copy span, in fact, it is even slightly better than the prediction resulting from the more detailed analysis. Why should more effort be spent to do the more detailed analysis? For design purposes, there is no reason to do the more detailed analysis. For purposes of developing a model of typing, it should be demonstrated that the mechanism of the model does not get in the way of good predictions. If situations occur where the entire mechanism of a model is not necessary for prediction, then the quicker predictions should be used, but more detailed analyses should not be substantially worse. The copy span task is one situation where considering only the model of working memory in the METT suffices to produce a good prediction. However, this simple analysis is not detailed enough to predict most of the Salthouse 29. For example, it would predict that the copy span is constant over skill. The more detailed analysis procedure, however, produces different predictions with a range of skills, reproducing the pattern observed in actual data (see Phenomenon 28, page 80).

Table 4-5: Copy span predictions for an 80 gwpm typist.

Character Typed	Available to Type	Copy span
O	ne_reason_is_	13
n	e_reason_is_	12
e	_reason_is_	11
_	reason_is_	10
r	eason_is_quite_	15
e	ason_is_quite_	14
a	son_is_quite_	13
s	on_is_quite_	12
o	n_is_quite_	11
n	_is_quite_	10
_	is_quite_	9
i	s_quite_	8
s	_quite_obvious;_	16

4.2.2.2 Phenomenon 14: Stopping span is between 1 and 2 characters

The stopping span, defined as the amount of material to which the typist is irrevocably committed to typing (Logan, 1982), averages only one or two keystrokes.

Logan (1982) measured this span directly by asking typists to stop typing as soon as they heard an auditory stop-signal. He did this in three slightly different experiments. The first experimental task, a time-contingent, discontinuous typing task, was as follows.

> Three-, five-, and seven-letter words were centered on the screen...The words were exposed for 1,000 msec, preceded by a fixation point that was exposed for 500 msec and was extinguished immediately before the word appeared. The intertrial interval was 2,000 msec and began as soon as the word was extinguished...The stop signal was a 500-msec, 900 Hz tone...It was presented at one of four delays (500, 650, 800, and 950 msec) following the onset of the word...subjects had no visual record of what they typed...The words within each length condition were balanced for hand repetition and alternation in the keystrokes they required... (Logan, 1982, p. 780)

To analyze this task, schedule charts were constructed for the perception and typing out of three- five- and seven-letter words with all possible same-hand, alternate-hand sequences. Imposed on top of these schedules was a perception of the tone (assumed to be 100 msec, estimated from a click-counting experiment, see Card, Moran and Newell, 1983, p.33) and a cognitive operator that recognized the tone to be the stop signal. The start of the perception operator was positioned at the start of the stop-signal (500, 650, 800 or 950 msec). The cognitive operator followed immediately and prevented any more characters from being initiated after that point in time. The number of characters typed after the stop-signal started was then the number of characters that had been initiated by the cognitive processor before the cognitive operator recognizing the stop-signal had begun. The motor operators associated with the initiation cognitive operators then completed the act by typing out the characters. The timeline form of the diagram shows the sequence of events most clearly (See Figure 4-13).

The average stopping span predicted by this analysis is 1.76 characters, 12.1% above the 1.57 characters observed by Logan. Although the METT is a course-grain model of typing and is intended to

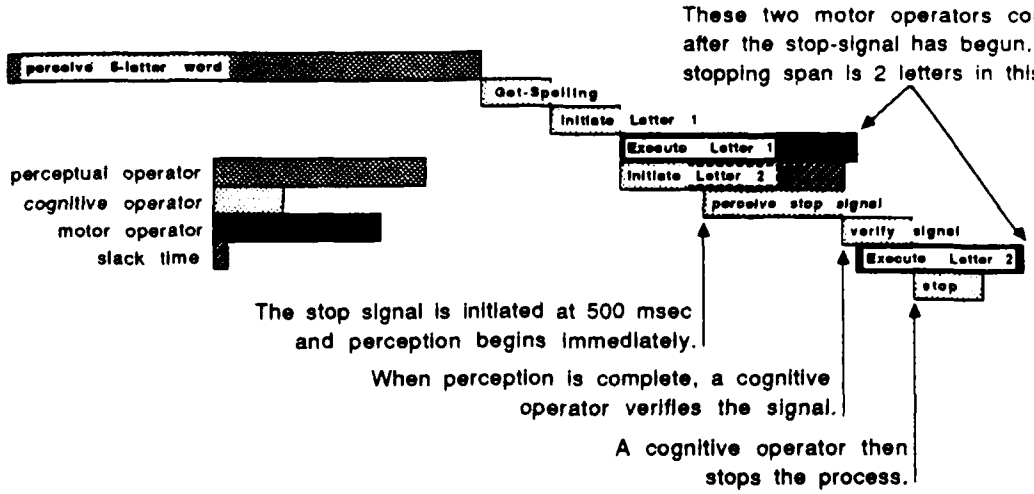


Figure 4-13: Timeline of a 60 gwpm typist, a stop-signal at 500 msec, and a word whose first two letters are on opposite hands and whose third letter is on the same hand as the second.

capture only first order effects, it is interesting to look more closely at the data and see which patterns the METT can reproduce. Logan reports several patterns in the data that suggest that typists do not type whole words before they stop, a behavior also not predicted by the METT.

> If they had typed whole words, the mean span should have decreased with stop-signal delay because fewer letters remain to be typed at the longer delays, and the spans should increase with word length because at each stop-signal delay, more letters remain to be typed with longer words. However, the data generally disconfirm these predictions. For five- and seven-letter words, the span increased with stop-signal delay instead of decreasing, and it did not increase with word length. For three-letter words, span increased from the 500-msec delay to the 650-msec delay but decreased at the longer delays when subjects had nearly completed the word before the signal occurred. This resulted in a lower mean span for three-letter words (1.43 letters) than for five- and seven-letter words. (Logan, 1982, p. 781)

Most of these patterns were predicted by the METT. For five- and seven-letter words the span did not decreased with delay, and did not differ with word length. For three-letter words, the METT did not predict the rise in span between the 500-msec and 650-msec delay, but the model did complete typing the word in many cases before the 950-msec delay, causing a substantially lower average span for that condition.

Two other patterns exist in the data:

> ...assuming that the span reflects the latency of the (internal) reponse to the stop-signal and that the latency of the response to the stop signal is constant over stop-signal delay...we would expect subjects to type fewer letters when the stop signal occurred at the 500- and 650-msec delays before subjects began typing...[Another pattern is that] the span appears to increase as subjects progress through the response sequence, although this effect did not replicate in [the other experiments]. (Logan, 1982, p. 782)

The first pattern did not appear in the METT predictions, span remained the same or decreased between the 500-msec and 650-msec delays. However, if Logan's explanation is correct, that people had not yet started typing before these delays occurred, the METT could not possibly have picked this up because the predicted latency between presentation of the word and being committed to typing the first letter was always one perceptual operator and two cognitive operators (to get the spelling and initiate the first letter), 440 msec, less than the 500 msec of the smallest delay. This indicates that the perceptual process may be the weakest part of the METT. Since the other pattern in Logan's data did not replicate, the fact that is not predicted with the METT speaks in the model's favor.

Logan's second experimental task made the stop-signal contingent on an event, the typing of a specific character, and was as follows.

> ...the same as in Experiment 1 except that the routine that accepted responses from the keyboard was rewritten to present a stop signal when a prescribed number of keystrokes had been registered (i.e., immediately after the *n*th keystroke). The copy to be typed was the five- and seven-letter words from the first experiment...The stop signal occurred on 20% of the trails...at one of four delays (after 1, 2, 3, or 4 keystrokes had been registered). (Logan, 1982, p. 782)

This event-contingent stopping task was analyzed in the same way as the time-contingent task, looking at all possible same- and alternate-hand sequences of five- and seven-letter words and all possible stop-signal onsets (after the first, second, third and fourth letters typed). The average predicted stopping span was 1.55 characters, 9.9% above the observed 1.41 characters. The observed results showed that

> ...the span was not substantially affected by word length or stop-signal delay except when signals occurred on the fourth letter of five-letter words. In this situation, only one letter was left to be typed. Clearly, there was no tendency to type whole words. (Logan, 1982, p. 783)

This pattern was reproduced in the predictions.

Finally, Logan's third experiment, also event-contingent, examined stopping behavior within a sentence rather than a single word. The experimental task was as follows.

> The copy to be typed was a set of 300 sentences of the form "the [noun] [verb]ed the [noun]," made from the five- and seven-letter words from Experiments 1 and 2. The nouns were either 5 or 7 letters long, but the verbs were 5, 6, 7, or 8 letters long because letters sometimes had to be added to make sense...There were 300 trials, and the stop signal occurred on 33% of them...at one of 20 positions in the sentence...Twelve of the positions were within words (the first, second, third, and fourth letters of the noun, the verb, and the second noun), and eight of the positions were between words (the last letter and the following space from the first four words of the sentence). (Logan, 1982, p. 784)

The same type of analysis was used for this task and the result was an average predicted stopping span of 2.08 characters, 3.7% below the observed average of 2.16 characters. In this experimental situation, the METT predicted fewer of the patterns. However, the pattern that Logan discussed at greatest length was predicted by the METT. The pattern is that "Spans before 'the' tended to include the word and the space following it". Logan cites this as evidence that "the" is typed ballistically and attributes it to the frequency of the word in the language and its frequency within the experimental situation as well. The METT suggests another explanation. The letters and the space in "the_" alternate hands, this allows the cognitive processor to initiate the whole word and the space in advance of the motor execution of the letters. Thus the stop signal often comes at a point where the letters have been initated, and must ballistically complete. This situation also exists with other frequent words, like "and_", and is an interesting prediction for future empirical verification.

Salthouse presents this as a quantitative phenomenon. However, examination of Logan's experiments allows us to upgrade it to a parametric phenomenon, because the different tasks produce different estimates of the stopping span (see Table 4-6). The METT predicts the pattern of results found by manipulating the tasks with an average absolute percent error of 8.6%.

This is a parametric phenomenon with a parametric METT prediction.

Table 4-6: Stopping span predictions for three different tasks.

Experiment Number	2	1	3
Signal Contingency	event	time	event
Context	single word	single word	sentence
Observed Stopping Span	1.41	1.57	2.16
Predicted Stopping Span	1.55	1.76	2.08
Percent Error	-9.9%	-12.1%	3.7%

4.2.2.3 Phenomenon 15: Eye-hand span is between 3 and 8 characters

The eye-hand span, defined as the amount of material intervening between the character receiving the attention of the eyes and the character whose key is currently being pressed, ranges between three and seven characters for average to excellent typists.

This phenomenon has been measured in two ways. Butsch (1932) recorded eye-movements while a person was typing and synchronized them with keying records to determine the position of the eye at each keystroke. He reported that the eyes are 4.9 characters ahead of the fingers, on average, for a group of typists averaging 60 wpm. The other method of estimating the eye-hand span is to use the restricted preview paradigm (see Section 4.2.1.6) and define the eye-hand span as the smallest number of characters in the preview window at which the typists reaches her asymptotic speed. Using this method, several researchers (Hershman and Hillix, 1965, Salthouse, 1984a, 1984b, 1985, and Shaffer, 1973) have reported between three and eight characters as the eye-hand span for moderately skilled typists.

Returning to the detailed examination of the restricted preview task in Section 4.2.1.6, the figure (Figure 4-8) indicates that the typist was observed to reached her asymptotic speed with a window between seven and nine characters in length (Salthouse assumes this to be an eight-character eye-hand span). The predicted curve is much smoother and actually does not reach normal typing speed until the 19-character window. This discrepancy can be partially accounted for by the difficulty in comparing the predicted results to the criterion used by the researcher to determine when asymptotic speed had been obtained.

The eye-hand span in Study 1 was defined as the smallest window at which the first quartile was greater than the second quartile of normal typing. This procedure effectively identified the span as the number of display characters at which 75% of the interkey intervals exceeded the median interval from normal typing.

A revised definition of reaching asymptotic speed with the predicted performance uses an estimate of the observed distribution. That is, the observed data indicate that the first quartile, is on the average, 19% smaller that the second quartile. Thus the predicted performance was taken as the median, or second quartile estimate, and the first quartile was estimated at 19% smaller than that value. That gives the predictions in Figure 4-7.

These predictions of the first quartile show that the first quartile becomes equal to the second quartile of normal typing (96 msec) at a nine-character preview window. Thus, a nine-character eye-hand span is predicted using this analysis, 12.5% above the eight-character eye-hand span reported for this subject.

This is a quantitative phenomenon with a quantitative METT prediction.

Table 4-7: Interkeystroke interval (msec) for different size preview windows for a 120 gwpm typist.

Preview Characters	Q1	Q2
unlimited	78	96
19	78	96
11	83	103
9	96	119
7	104	128
5	138	170
3	200	247
1	369	456

4.2.2.4 Phenomenon 16: Eye-hand span decreases with decreasing meaning

The eye-hand span is smaller for unfamiliar or meaningless material than for normal text.

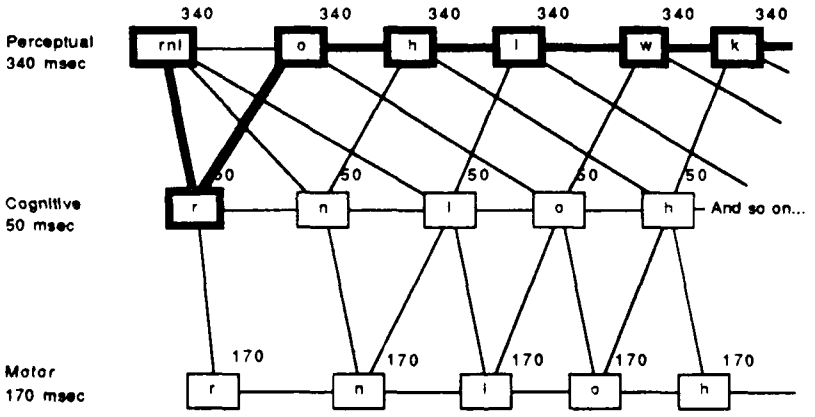


Figure 4-14: Portion of a schedule chart for typing random letters.

The METT analysis of this phenomenon is obtained from the critical path on the schedule chart for typing random letters with different preview windows. With infinite preview (Figure 4-14), the critical path is dominated by the perceptual processes, and will not change at all until the window is at 2-letters. At a 2-letter window, one motor operator appears on the critical path because the window must be advanced by the typing of a character. This one slight change increases the average interkey interval by only 3% (Figure 4-15). When the preview window is decreased to 1-letter, then the critical path changes drastically, becoming completely serial, every operator dependent on the completeion of every previous operator (Figure 4-16). The average interkey interval increases by about 64% over the 2-letter window interval. The quartile distribution reported by Salthouse again puts the first quartile 19% below the second quartile for this random-letter condition. Assuming this distribution, the predicted eye-hand span is somewhere between one and two letters; predict 1.5 letters given no other information. Comparing this to the reported 1.75 characters, there is a 14.3% absolute error between predicted and observed eye-hand span for typing random letters.

This is a quantitative phenomenon with a quantitative METT prediction.

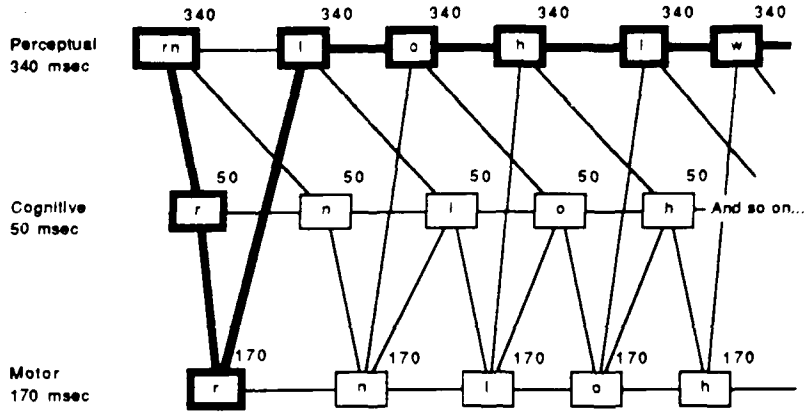


Figure 4-15: Portion of a schedule chart for typing random letters with a 2-letter preview window.

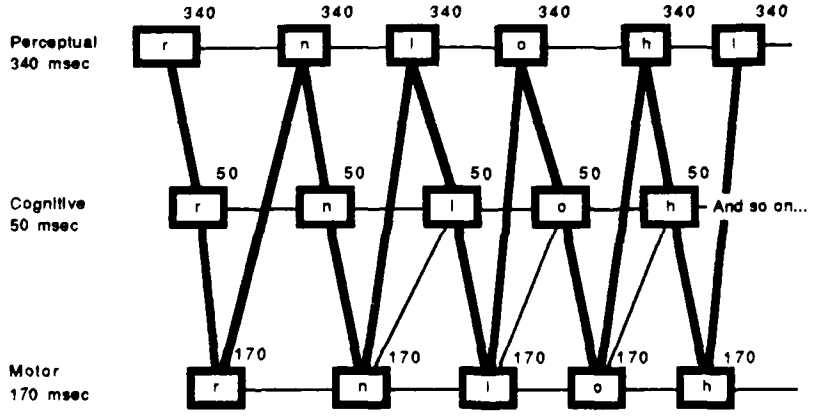


Figure 4-16: Portion of a schedule chart for typing random letters with a 1-letter preview window.

4.2.2.5 Phenomenon 17: Replacement span is about 3 characters

> Typists appear to commit themselves to a particular character approximately three characters in advance of the current keystroke.

The replacement span is the number of characters between the character currently being typed and where a change can be made in the copy, which the typist will be able to detect and typed as changed. Salthouse and Saults (1987) (Salthouse & Saults, 1987) measured the replacement span as follows.

> ...the replacement span is defined as the key-stroke-replacement interval corresponding to a .5 probability of typing the second [changed from what was originally there] character...Salthouse and Saults (1987) found the replacement spans to average 2.8 and 3.0 characters in two studies...

The task given to the typists was:

> In this task the material will always be lower case, but on some occasions a letter will be changed in the display. You should ignore the original character when this happens and type the 'corrected' version that appeared most recently on the display. Remember to try to type exactly what appears on the screen in as normal a fashion as possible. (Salthouse & Saults, 1987, p. 189)

An analysis of the example sentence laid out for a 60 gwpm typist (approximately the average typist in

the study) was done to simulate this task. The task timeline form of the diagram was used. Excluding the first three and the last 13 letters (the first three arbitrarily because I wanted the typing to get going before switching characters and the last 13 because those characters were beyond where there was anything more to look-ahead and see), each letter was examined to find out when the change had to occur in order for the typing of that letter to be stopped, and changed to the new letter. For a letter to be stopped, a cognitive operator recognizing a stop signal had to be started before the cognitive operator that initiates the typing of the letter was started. For this cognitive operator to begin, there would have had to be a perceptual operator (perceiving that a change in the display had occurred) started and completely finished. This perceptual operator did not have to perceive the exact nature of the change, just that a change had occurred. One hundred msec was used as the duration of this "perceive-a-change" perceptual operator, because it is a typical perceptual operator (Card, Moran and Newell's Middleman perceptual processor cycle time). This point in time (see Figure 4-17) was the point at which the change had to occur in the stimulus for the typist to make the change in her typing, with one important constraint. If the perceptual processor was busy elsewhere (looking ahead at the copy) then the change would not be noticed, so the change would actually have to occur before the perceptual processor looked ahead. This constraint is meaningful within the model; the processors work serially within themselves. It is also meaningful within a detailed task analysis; if the person is looking ahead, then she might not notice a change in a different part of the display. After finding the necessary starting point of the change in the display, a count was made of the motor operators that would complete themselves in between this starting point and the point that the changed-letter was to have been typed. Because the METT has no mechanism for figuring the probabilities of behavior, this count is used as the replacement span. With this analysis, the replacement span was predicted to be 2.1 characters, 27.6% below the observed 2.9 characters.

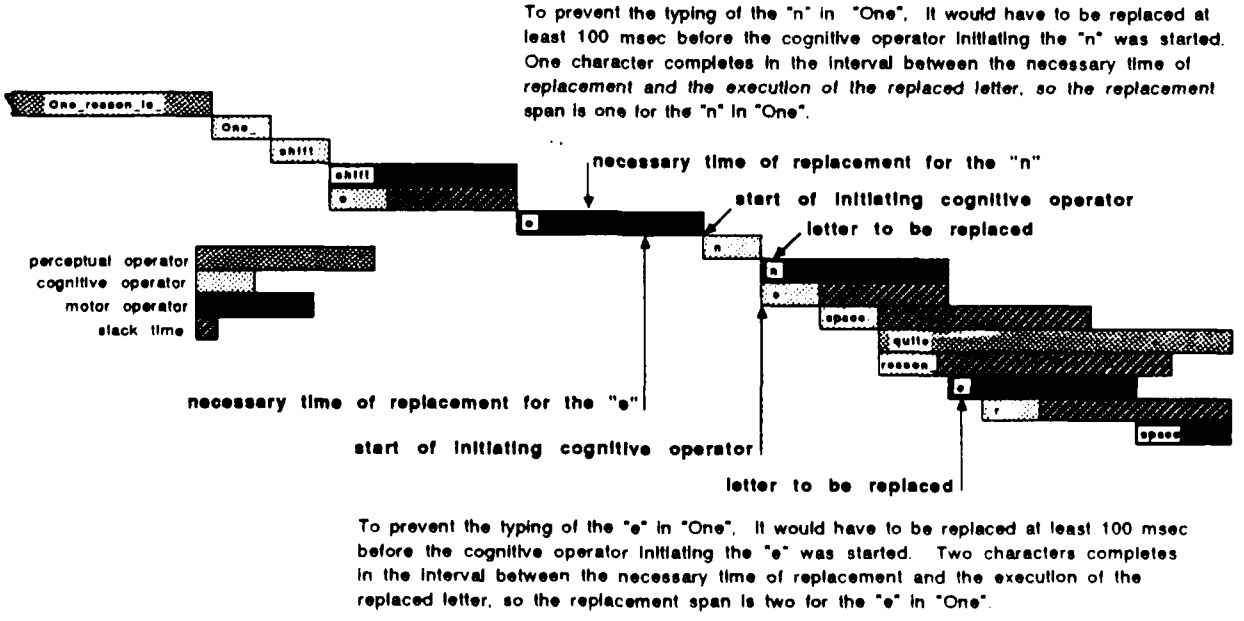


Figure 4-17: Portion of a task timeline for an 60 gwpm typist showing the replacement span analysis.

The analysis of the replacement span is very similar to the analysis of the stopping span. The

difference between the two tasks is that in the stopping span task, the perceptual process can be completed before the stop-signal occurs, whereas in the replacement span task, perception of the to-be-typed copy occurs throughout the task. The stopping span predictions are very good, even parameteric with slightly different tasks, and the replacement span prediction is not as good. This indicates that the model of perception is probably the weak point of the METT. However, qualitatively, the replacement span is predicted to be positioned between the stopping span and the eye-hand span, as it occurs in the observed data. Therefore, this is a quantitative phenomenon with a qualitative METT prediction.

4.2.3 Errors

The next five phenomena deal with errors in typing. In its current form, the METT makes only response-time predictions, and cannot make predictions about the types of errors that will be made. Therefore, all these phenomena are not covered by the METT. However, the error phenomena are discussed in terms of their implications for an extension to the METT. Such an extension might be similar to Rumelhart and Norman's motor response system, differentiating between fingers, making movements dependent on surrounding context, and adding noise to the system. Any extension that is proposed should address the five errors described in Phenomena 18 through 22.

4.2.3.1 Phenomenon 18: Only a fraction of errors are detectable without reference to the typed copy

Only between 40% and 70% of typing errors are detected without reference to the typed copy.

This phenomenon involves self-monitoring of the processes that produce the typing behavior. As Salthouse details, errors can have their source at many different points in the process. The METT could attribute errors to faults in the perceptual processor (perceiving or encoding a word incorrectly), the cognitive processor (perhaps remembering the spelling of a word incorrectly), or the motor processor (executing the motions to hit a specified character incorrectly).

The current METT does not have a model of the feedback necessary within each processor to detect errors as they are occurring, or of how the processors monitor each other's performance. Thus, this phenomenon is beyond the scope of the current model. However, information about the distribution of detectable errors could help shape extensions to the METT.

Rabbit (1978) reports the types of errors detected when typists cannot see what they have typed (Table 4-8. In his terminology, "compound errors" refers to when "two or more letters of text might be transposed or garbled in order, or an incorrect keystroke, which had no relation to the text, might be followed by one or more other incorrect keystrokes, by a transposition of two letters, or by an omission of one or more letters". (p. 949) Unfortunately, the compound and single-letter categories cannot be mapped into the substitution, intrusion, and transposition errors reported elsewhere and discussed by Salthouse. Thus, the only clear conclusion available from the Rabbit results is that omission errors have a very low probability of detection. In addition, Rabbit states (though without numeric support) the fact that the typists "made compound errors due to 'de-referencing' of hands with respect to the keyboard, and the probability of detection of such errors was, not surprisingly, very low." These two results together indicate that proprioceptive feedback does not give useable information about the force with which a key is hit (too little force produces an omission error, see Phenomenon 21), or about the position of the hand relative to the keyboard. However, useable feedback about the relative position of the fingers as they hit the keys might lead to the possible higher detection rate of substitution, intrusion, and transposition errors. Thus, it

seems reasonable that the next level of detail that needs to be added to the METT in order to pick up this phenomenon would introduce feedback about the position of the fingers relative to each other, but not information about the force of a stroke or the position of the hand.

This is a quantitative phenomenon not covered by the METT.

Table 4-8: Errors made and detected in Rabbit's experiment (1978).

	Errors Made	Errors Corrected	Percentage Corrections
Grand Total of all Errors	7089	4447	62.73%
"Compound Errors" involving two or more responses	3403	1965	57.74%
Single letter mistypes	3591	2478	69.00%
Omission Errors	95	4	4.21%

4.2.3.2 Phenomenon 19: Substitution errors are mostly adjacent keys

> Many substitution errors involve adjacent keys...Results from highly skilled typists indicated that from 31% to 59% of substitution errors involved horizontally adjacent keys, and between 8% and 16% involved vertically adjacent keys.

This type of error, and those in the four immediately following phenomenon, seem to originate in the motor processor. The cognitive processor has sent a command to the motor processor initiating the typing of a character, and it appears to be the correct character at that point (or the substitution would have been with a character in a different region of the keyboard). The motor processor, therefore, is at fault in the execution of motion to carry out the typing of that character. A more detailed model of the motor processor might include noisey specifications of the position of the key to be hit or the current position of the finger, producing incorrect keystrokes on some occasions.

This is a quantitative phenomenon not covered by the METT.

4.2.3.3 Phenomenon 20: Intrusion errors are mostly short interkey interval

> Many intrusion errors involve extremely short interkey intervals in the immediate vicinity of the error...interpreted as being caused by the nearly simultaneous contact of two adjacent keys by a finger imprecisely positioned above the target key...the median ratios [are] considerably less than 1.0 for the error keystrokes [0.68]...and the immediately following keystroke [0.87]...

Since the finger has been positioned almost correctly (with either the position above the key being about a key radius off-center, or the vertical trajectory not being at the correct angle), the correct letter has probably been initiated by the cognitive processor. Thus, as in Phenomenon 19, the motor processor is probably to blame for the error, and noisey specifications might account for the fault.

This is a quantitative phenomenon not covered by the METT.

4.2.3.4 Phenomenon 21: Omission errors are mostly long interkey interval

> Many omission errors are followed by a keystroke with an interval approximately twice the overall median...consistent with insufficient depression of the keystroke for the omitted character such that its latency is incorporated into the interval for the following keystroke.

In this situation, the finger has been positioned correctly above the letter to be typed, but the force with

which the key is hit is too small. Again, this suggests that the letter was initiated correctly by the cognitive processor, but executed incorrectly by the motor processor. As in Phenomena 19 and 20, a detailed motor program model would explain the specification of force to be used on a key and how and why it might be specified or executed incorrectly.

This is a quantitative phenomenon not covered by the METT.

4.2.3.5 Phenomenon 22: Transposition errors are mostly cross-hand

> Most transposition errors are cross-hand rather than within-hand...The percentage of total transposition errors that involved fingers on opposite hands reported by Grudin was 78%, compared with a chance value...of approximately 53%.

Again, a detailed model of the motor programs might shed light on how the sequencing of two signals might become confused. A clue to the mechanism has to do with the timing of the incoming "initiate-letter" signals from the cognitive processor. Since most transposition errors are cross hand, most of the initiation signals from the cognitive processor overlap the motor programs for the previous letter in the motor processor. Within the current METT, this timing is the critical difference between the opportunities for error and the situations where transposition errors rarely occur.

This is a quantitative phenomenon not covered by the METT.

4.2.4 Skill Effects

The last seven phenomenon deal with effects of skill in typing. The METT is a model only of *expert* transcription typing, so the skill effects are examined only in the range above 60 gwpm. Some of the skill effects are still present in this range, but less pronounced than with the full range of novice through expert typists.

4.2.4.1 Phenomenon 23: 2-finger digrams improve faster than 1-finger digrams

> Digrams typed with two hands or with two different fingers of the same hand exhibit greater changes with skill than do digrams typed with one finger.

The reported decreases in interkeystroke interval vary. Salthouse (1984a) reports rates of -2.08 msec/nwpm for two-hand digrams, -2.38 msec/nwpm for two-finger digrams, -1.91 msec/nwpm for one-finger digrams, -0.85 msec/nwpm for one-letter digrams.

The METT does not cover this phenomenon because it does not differentiate between fingers. However, it does make an interesting prediction about the improvement of same-hand and alternate-hand digrams. For a 60 gwpm typist, the alternate-hand interkey interval is 170 msec (one motor operator) and the same-hand interkey interval is 220 msec (one motor plus one cognitive operator). For a 120 gwpm typists the interkey intervals are 70 msec and 120 msec, respectively. This gives an absolute reduction in interkey interval of -1.67 msec/gwpm for both alternate- and same-hand digrams. Thus, the METT predicts the magnitude of the dependence on skill and that there is no difference between the rates of change of alternate- and same-hand digrams. This prediction is matches the data reported by Salthouse if you average the three same-hand improvement rates. Then the rate of change for alternate-hand digrams is -2.08 msec/nwpm (19.7% error) and -1.71 msec/nwpm for the same-hand digrams (2.3% error). The prediction of no-difference made by the METT is reflected in the less than 20% difference between the two observed rates of change.

This is a quantitative phenomenon not covered by the METT.

4.2.4.2 Phenomenon 24: Tapping rate increases with skill

The rate of repetitive tapping is greater among more skilled typists.

Increase in typing skill decreases the motor operator for hitting a key (motor operator duration/skill interaction assumption). It is assumed that the motor operator for the similar task of hitting a single key in repetitive tapping would also decrease (operator similarity assumption). Thus, the speed of the tapping would increase with typing skill. However,no rates of change of tapping interval are available given in the literature to test quantitative predictions.

This is a qualitative phenomenon with a qualitative METT prediction.

4.2.4.3 Phenomenon 25: Variability decreases with skill

The variability of interkey intervals decreases with increased skill of the typist. At least two types of variability can be distinguished in typing, and both have been reported to be smaller among fast typists. One type is interkeystroke variability, in that it refers to the distribution of interkey intervals across different keystrokes and different contexts. The second type of variability is intrakeystroke variability or repetition variability. This is the distribution of interkey intervals for the same keystroke in the same context, but across multiple repetitions.

In as much as the METT does not provide the detail needed to predict the latter variability, it also does not make any predictions concerning intrakeystroke variability changing with skill. However, with interkeystroke variability, a strong prediction comes from the same-hand constraint assumption (Assumption 2b) and the motor operator duration and skill interaction (Assumption 6c). The same-hand constraint assumption implies that the difference between a same-hand keystroke and an alternate-hand keystroke will be the time of a single cognitive operator, 50 msec. The motor operator duration and skill interaction assumption implies that this will be a constant difference across the range of skill. Thus, the METT makes the prediction that the interkeystroke variability, when considering same-hand and alternate-hand keystrokes, will not decrease with increasing skill.

This prediction is contrary to the phenomenon as stated by Salthouse. However, closer examination of the data indicates that the decrease in variability is primarily due to same-finger digrams (Salthouse, 1988; Gentner, 1983). The METT does not differentiate between fingers on a single hand, so it does not make a prediction about the variability within a hand or how it changes with skill. The prediction that the variability between same-hand and alternate-hand keystrokes will be constant across skill and about 50 msec finds strong support in the data reported by Ostry (1983) (Figure 4-18). He shows a constant difference between same-hand and alternate-hand keystrokes of about 45 msec (giving an error of 11.1%). Given the good agreement between data and prediction for the type of variability the METT can address, this will be considered a successful quantitative prediction.

This is a quantitative phenomenon with a quantitative prediction.

4.2.4.4 Phenomenon 26: Eye-hand span increases with skill

The eye-hand span is larger with increased skill.

The eye-hand span is the minimum number of characters ahead of the character being typed that the eyes must be to achieve normal typing speed. Using the analysis technique described in Section 4.2.2.3, the eye-hand span was calculated for a 120 wpm typist examined in that section and for 60 wpm and 90 wpm typist (Table 4-9). Assuming the same distribution as the observed data, with the first quartile being

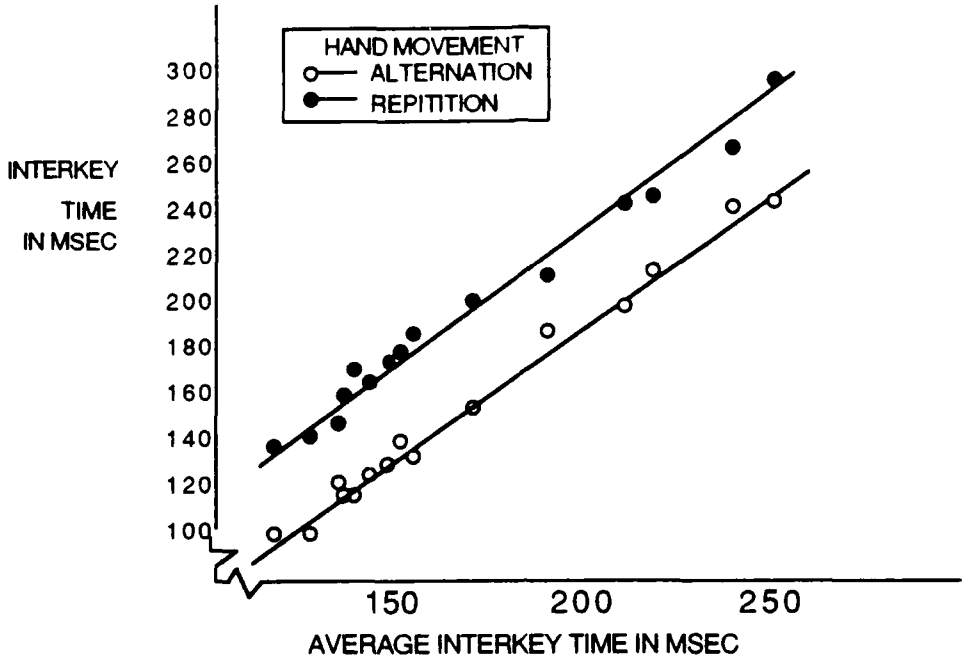


Figure 4-18: Average interkey times as a function of typing speed (Ostry, 1983).

19% smaller than the second quartile, then the three-character preview window condition is the first condition where the interkeystroke interval of the first quartile exceeds the second quartile interkeystroke interval of the infinite window condition. Therefore, the eye-hand span would be set at 4 characters for 60 gwpm typists, at 8 characters for a 90 gwpm typist, and at 9 characters for a 120 gwpm typist. The best fit line to these results has a slope of 1 character per 12 wpm increase in speed (0.083 characters per gwpm speed-up). Salthouse reports slopes between 0.025 and 0.060 characters per wpm speed-up, the predicted result being 38.3% above the high end of this range.

This is a quantitative phenomenon with a qualitative METT prediction.

Table 4-9: Interkeystroke interval (msec) for different size preview windows for 60, 90 and 120 gwpm typists.

	60 gwpm		90 gwpm		120 gwpm	
Preview Characters	Q1	Q2	Q1	Q2	Q1	Q2
unlimited	158	195	109	135	78	96
19	158	195	109	135	78	96
11	159	196	112	138	83	103
9	170	210	124	153	96	119
7	177	218	133	164	104	128
5	190	234	156	193	138	170
3	247	305	215	266	200	247
1	450	556	402	496	369	456

4.2.4.5 Phenomenon 27: Replacement span increases with skill

The replacement span, indicating how far in advance of the current keystroke the typist commits to a particular character, is larger among more skilled typists.

Using the analysis technique described in Section 4.2.2.5, the replacement span was predicted for three typing speeds within the expert range (60 gwpm, 90 gwpm, and 120 gwpm). The replacement spans were 2.1 characters for the 60 gwpm typists, 2.5 characters for the 90 gwpm typists, and 3.2 characters for the 120 gwpm typists, showing an increase of about 1 character for every 60 gwpm increase in speed. This slope is half the 1 character/30 wpm speed increase reported by Salthouse. The original paper reporting this results (Salthouse & Saults, 1987) does not show a scatter plot of the observed performance, so it is not possible to tell whether the slope flattens out at the expert range of the skill dimension. This is another prediction of the METT left for future empirical verification.

Given the difference between predicted and observed slopes, this is a quantitative phenomenon with a qualitative METT prediction.

4.2.4.6 Phenomenon 28: Copy span is dependent on skill

The copying span is moderately related to typing skill.

The copy span is the maximum number of characters beyond the character being typed that a typist looks in the course of normal typing. Using the analysis technique described in Section 4.2.2.1, the copy spans for the average typists in each of the two expert speed ranges (between 60 and 80 nwpm, M=72.5 nwpm, and over 80 nwpm, M=90.6 nwpm) were calculated. For 70 gwpm typists, the copy span is 12.2 characters, 21.3% from the observed 15.5 characters. For 90 wpm typists, the copy span was 11.7 characters, 14.0% from the observed 13.6 characters, for an average error of 17.7% for the two predictions.

Copy span is observed, and predicted, to be dependent on skill within the expert range. However, the copy span goes down with skill rather than up, contrary to the general trend across the whole range of novice to expert typists. The specific mechanism for this trend within the METT is a complex relationship between the durations of different operators, observable through analysis of detailed timeline diagrams. The intuitive statement of the meaning of this downward trend in the expert range is that as skill increases, not as much need be kept in working memory to maintain high levels of speed.

This is a parametric phenomenon with a parametric METT prediction.

4.2.4.7 Phenomenon 29: Stopping span increases with skill

Fast typists have larger stopping spans than slow typists.

Logan (1983) reports no significant relationship between stopping span and skill. However, in a later experiment with more subjects, Salthouse & Saults (1987) report a positive correlation between stopping span and typing skill. The METT prediction of this relationship was examined by calculating the stopping spans for typists at 60 gwpm, 90 gwpm, and 120 gwpm for Logan's event-contingent experimental task (see Section 4.2.2.2, p. 70). The stopping spans were 1.55 msec, 1.55 msec and 1.76 msec, respectively. Thus, the METT does not predict a strong increase in stopping span with skill, although a slight net increase is indicated. Salthouse & Saults do not report the slope of the relationship (just the correlation coefficient), making this a qualitative phenomenon.

Thus, this is a qualitative phenomenon with a qualitative METT prediction.

4.2.5 Summary of the METT Account of the Salthouse 29

The Salthouse 29 phenomena range from being qualitative in nature (e.g., Phenomenon 3: There is no relation between typing skill and degree of comprehension of material that has been typed.), to being quantitative and even parametric (e.g., see Figure 4-8 in Phenomenon 6, restricted preview). Likewise, the METT makes predictions that agree parametrically, quantitatively, or qualitatively with the reported phenomena. Table 4-10 gives the totals of the different types of accounting the METT makes of the data. A summary of the way in which the METT accounts for the Salthouse 29, by individual phenomenon, is given in Figure 4-23, page 86.

Table 4-10: Phenomena accounted for by the METT.

Type of Prediction	Type of Phenomenon		Number
Parametric	Parametric		3
Quantitative	Quantitative		7
Qualitative	Qualitaive		6
Qualitative	Quantitative		5[9]
Not covered			8
Of all 29 phenomena:			
Total as good as the data		16/29	55%
Total accounted for at least qualitatively		21/29	72%
Of the 21 phenomena covered by the METT:			
Total as good as the data		16/21	76%
Total accounted for at least qualitatively		21/21	100%

4.3 Beyond the Salthouse 29

The Salthouse 29 provide a comprehensive list of phenomena associated with transcription typing. But of course, these are not the only phenomena that exist, or that are even important (although the others may not be documented well enough to satisfy Salthouse's criterion of robustness). Obvious and important phenomena that are missing include the shape of the learning curves, the speed-accuracy trade-off relationship, and the transfer to other keyboards. These and other new phenomena should be treatable by the METT, as indeed it should treat any idiosyncratic typing task that arises in the design of human-computer interfaces.

Just for the joy of it, we treat two more phenomena: the detection span, a measure of how far in advance a typist can detect an oddity in the text, and the shape of the learning curves.

4.3.1 Detection Span

Late in the work of this thesis, we came across a just published paper by Salthouse and Saults (1987) which defined a new span characterizing typing behavior, the detection span. The instructions for the task measuring the detection span were as follows.

[9] If the Fitts Law extension is made as described in Section 4.2.1.11, page 60, then Phenomenon 11 becomes a qualitative prediction on quantitative data, raising that category to 6 and the total accounted for to 76%.

> In this task a large number of characters will always be visible on the display, but occasionally a capital letter will appear. Whenever you notice a capital letter anywhere on the line you should press the '/' key as soon as you can and then resume typing. The capital letters should not be typed as capitals, but whenever you detect an upper-case letter you should press the '/' key. Always try to type as normally as possible. (p. 189)

Using this task, with a stimulus material of randomly arranged four-letter words, the "detection span was defined as the median number of characters intervening between the target and the character currently being typed." However, the mean of the detection span was also reported and that is the measure the METT predicts. The observed mean detection span was 8.1 characters (SD = 4.6 characters) for one study and 7.8 characters (SD = 5.0 characters) for another. Both sets of participants had an average typing speed of about 60 gwpm.

The distribution of the observed detection spans were quite flat. Salthouse and Saults offer this interpretation of that result:

> The magnitude of the detection span was quite variable across subjects, possibly because several different strategies could be used in this task. On the one hand, subjects could simply try to type normally and emit a detection response only when a target was accidently encountered near the occurence of one's current keystroke. On the other hand, the subject could periodically decide to interrupt his or her typing to scan for targets, thus detecting the target at very great distances and resulting in larger detection spans.

This multiple strategy approach is inherent in the METT, and three different algorithms were assumed in order to make the detection span prediction.

The first algorithm (called the *spelling algorithm*) corresponds to Salthouse and Saults' first strategy. The algorithm assumes that the perception operator includes an encoding of whether a capital exists in the spelling of the word. When the spelling of the word is brought into working memory in preparation for that word being typed, a test is performed on the spelling to see if a capital exists. If a capital does exist, then the result of the test is to type the '/'. A critical path analysis of this algorithm (Figure 4-19) reveals that, since the intiation of the preceeding space triggers the retrieval of the spelling of the next word, the space is always typed before the '/' is hit. Since the letters of the word containing a captial cannot be typed until the '/' is hit, then the position of the capital letter in the word is the detection span (i.e., if the capital is the first letter, the detection span is one; if it is the second, the detection span is two, etc.). With a stimulus material of only four-letter words, this gives detection spans of 1, 2, 3, and 4 characters.

The second algorithm (the *perception algorithm*) has two versions: (*wait* and *parallel*. The assumption is made that the participants obeyed the instructions to "Always try to type as normally as possible", and did not stop and look ahead of where they would look in the course of normal typing, but made the judgement about whether a capital existed after the word was perceived normally. It is assumed that the perceptual operator encoded whether a word included a capital independent of its spelling. The test for a capital was performed as soon as the perception was complete, and the '/' hit if a capital was detected. The difference between the two versions of the algorithm is that in the wait version, a perception of a word is initiated in the course of normal typing, but the cognitive processor waits for the perception to be complete and makes the test before continuing with typing. Thus, this wait makes the perception-wait algorithm display behavior similar to the stop-and-scan strategy that Salthouse and Saults proposed. In the parallel version, the perception proceeds as it does in normal typing and the cognitive processor continues with the typing task until the perception is complete, and then the test is made.

In the wait case, a critical path analysis (Figure 4-20) reveals that since the initiation of a space at the end of a word is the cognitive activity that triggers the perception of the next word, and the decision about

the capital waits for the perception to be completed, then the first character of the word two words before the word with the capital will always be the character at the leftmost side of the screen and the detection span will be 10 plus the position of the capital within the word (i.e., 11, 12, 13, or 14).

In the parallel case, the critical path analysis (Figure 4-21) is more complex because the same-hand/alternate-hand pattern of letters influences how many characters are typed during the perceptual process. The detection spans predicted using this algorithm range from 7 to 13 characters.

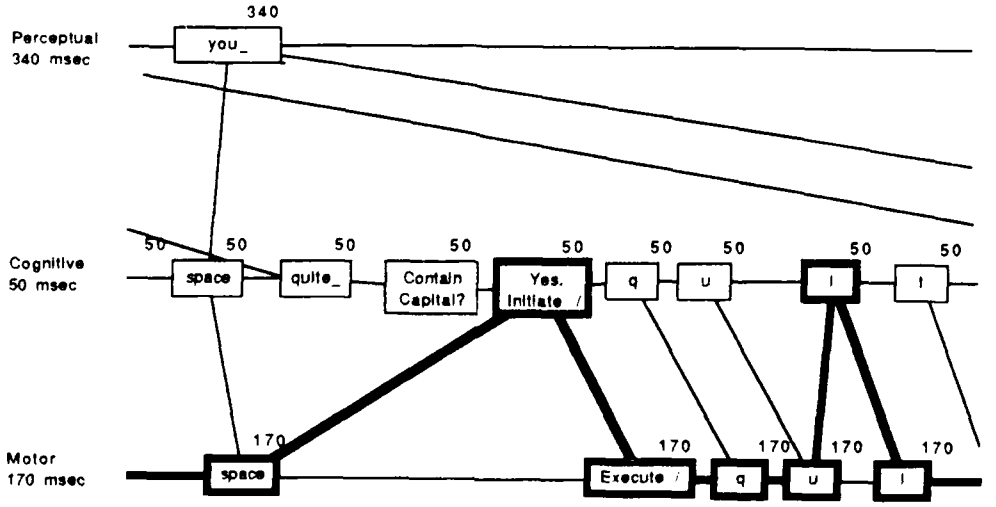


Figure 4-19: Schedule chart of the spelling algorithm for the detection span task.

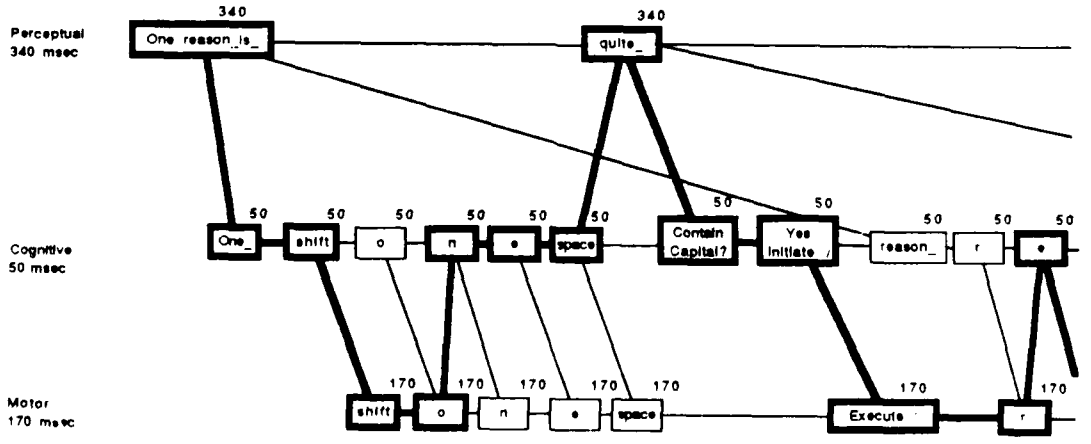


Figure 4-20: Schedule chart of the perception-wait algorithm for the detection span task.

Averaging over these possible algorithms, yields a predicted detection span of 8.56, 7.7% away from the 7.95 average observed by Salthouse and Saults. It is also worth noting that the extreme detection spans predicted are 1 and 14, whereas the extremes observed are 1 and 15. This result suggests that a true stop-and-scan strategy, like the one proposed by Salthouse and Saults, that assumes a violation of the task instructions by the participants, is not necessary to explain the data.

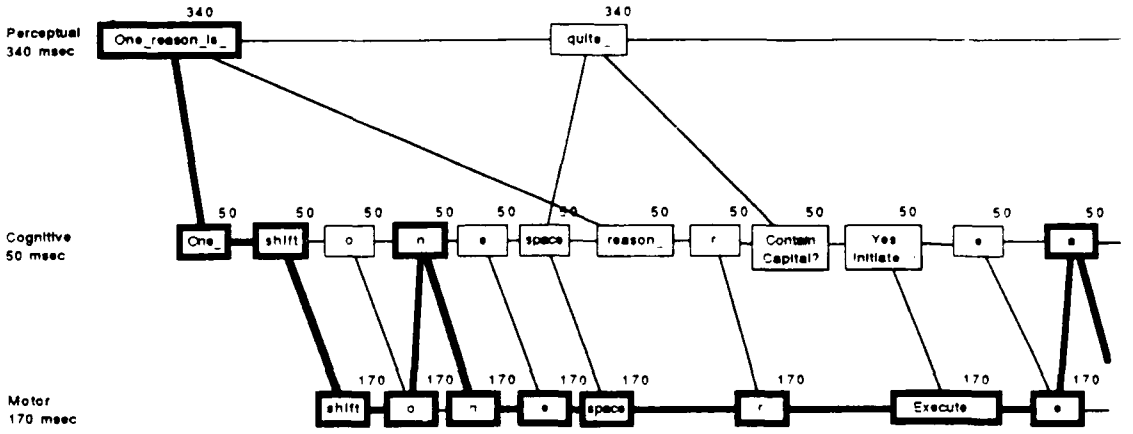


Figure 4-21: Schedule chart of the perception-parallel algorithm for the detection span task.

4.3.2 The Shape of the Learning Curve

As it stands, the METT is a model expert typing that predicts performance of a typist at a particular point of skill; it predicts neither the shape of the learning curve as a typist progress from novice to expert, nor the shape of the curve as an expert increases in speed. However, if the model were to be extended to make such predictions, it would have to abide within the MHP architecture. From the power law of practice (*Principle of* Operation 6), the MHP predicts that as typists improve from novices to experts, the shape of the learning curve will follow a power law. Gentner (1983) collected longitudinal performance data from typing students from the fourth through eighth week of a beginning typing class. He separated the keystrokes into four classes, 1-finger doubles (the same key hit twice in succession), 1-finger non-doubles (the same finger is used to hit two different keys in succession), 2-finger digraphs (two fingers on the same hand hit two different keys in succession), and 2-hand digraphs (two fingers on opposite hands hit two different keys in succession), and plotted the median inter-keystroke interval vs. the week of the class (Figure 4-22). The decrease in median inter-keystroke interval was well described by a power law, as predicted by the MHP.

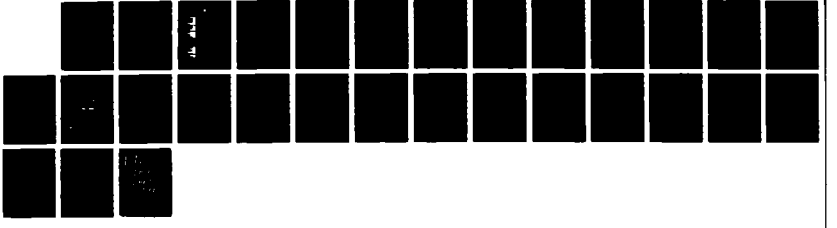
The power law of practice not only fits the data well, but also allows for more detailed interpretation of the data. For instance, Gentner observes the pattern in the different learning rates (slopes) for the different classes of key sequences. By comparison to the performance of typists at different skill levels, and to expert typists asked to type at different speeds, he concludes that the longitudinal data mirror the cross-skill and speed/accuracy tradeoff data and that these results suggest that the mechanism underlying the speed changes during skill acquisition is the same mechanism underlying intentional speed changes of an expert.

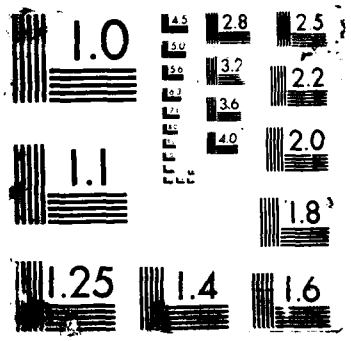
4.4 Discussion

The purpose of this chapter was to explore the predictive power of the GOMS model of reaction-time tasks developed within the domain of SRC in a new domain: expert transcription typing. The basic form of the model and two parameters (time to perceive a written word, syllable or letter and the cognitive cycle time) were transported from the SRC domain to the typing domain. A few task-specific assumptions were added to tune the model for typing tasks. The motor operator parameter for expert typists of different

AD-A195 590 CONTRIBUTIONS TO ENGINEERING MODELS OF HUMAN-COMPUTER INTERACTION VOLUME.. (U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF PSYCHOLOGY B E JOHN 06 MAY 88 2/2

UNCLASSIFIED 1-51206A-VOL-1 N00014-87-K-0432 F/G 12/8 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

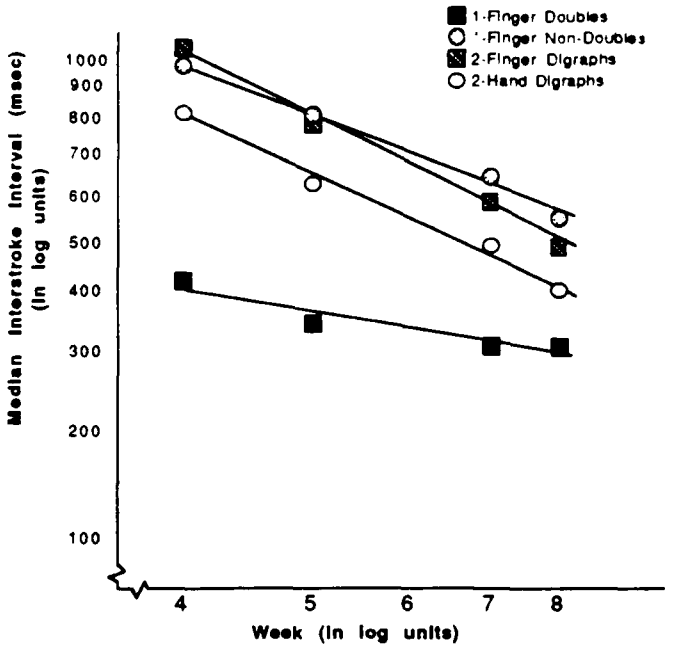


Figure 4-22: *Learning curves for a single typist from Gentner, 1983.*

average typing speed was calculated from performance on a standard typing test. Several other time parameters (e.g., perceiving an auditory tone) were estimated from previous experiments in other domains (e.g., the fusion of clicks experiment, Cheatham and White, 1954). From these a priori definitions and estimates, the METT was used to analyze the Salthouse 29 phenomena of typing and several other typing observations.

As summarized in Section 4.2.5, the METT makes good predictions on the Salthouse 29 phenomenon. It continues in that vein by making a prediction of the detection span (Section 4.3.1) to within 8% of the observed data. On a more global level, the shape of the learning curve for the acquisition of typing skill conforms to that predicted by the MHP (Section 4.3.2). Figure 4-23 shows the overall performance of the METT on all the phenomena discussed.

These results lead to the conclusion that the METT is an excellent example of an engineering model. It makes zero-parameter quantitative predictions almost always to well within 20% of observed behavior. It's level of approximation can be varied in accordance to the task being examined. It covers a broad range of typing tasks, more than any other quantitative typing model in existence. The six assumptions that specify the model are simple and objective in their application, suggesting that the METT would be an easy model for computer designers to learn and use. In addition, the METT is not a stand-alone model of typing, but a fully integrated part of the MHP, derived from the structure of GOMS and the assumptions of the MHP with minor supplemental assumptions to suit the task domain, and indeed, the success of the METT provides support for the MHP architecture as a whole.

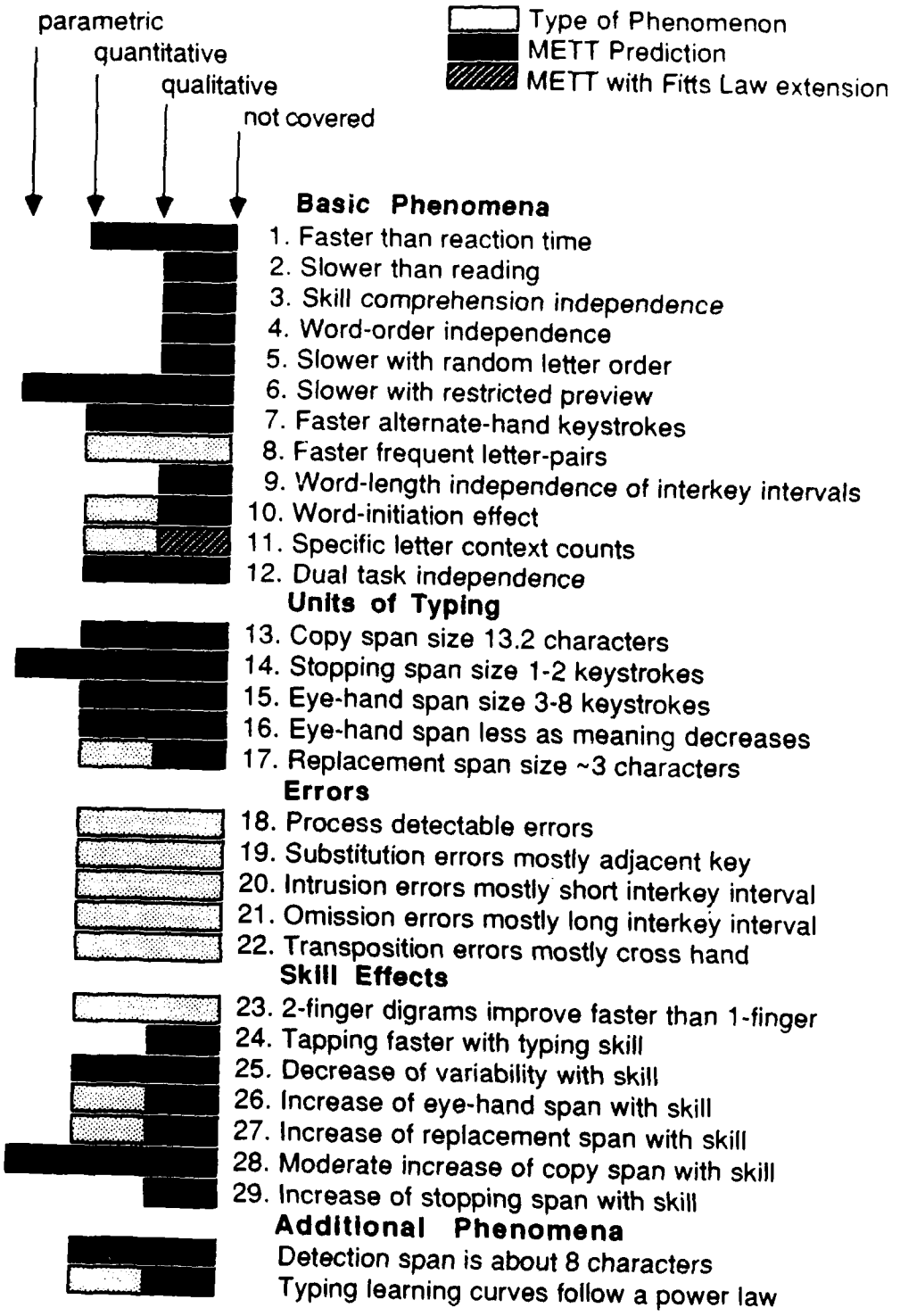


Figure 4-23: Summary of the METT account of the Salthouse 29.

5. Contributions of this Dissertation to HCI

The two previous chapters have provided models for two domains within HCI. The quality of those models was assessed in their respective chapters. They each fill a gap in the theory in their respective domains. However, the goal of this dissertation is broader than simply providing a model in a narrow domain or two. The goal is to contribute to engineering models that explain the behavior of people as they perform tasks when interfacing with computers. The focus has been on behavior at the interaction-level of HCI - immediate behavior at the interface. This chapter attempts to assess the success of this dissertation in terms of the broader goal.

The assessment will be made by examining how the component subdomain contributions fit into the total picture. This requires three things: 1) a list of the activities involved at the interaction level of the human-computer interface, 2) a review of the current state of model development within HCI, and 3) a positioning of the current work.

5.1 Activities at the Interaction Level of HCI

The human/computer system is comprised of a user, a computer system (including documentation or other information sources), an interface between the two, and a goal to drive the interaction. This system is used to accomplish tasks like writing a paper, creating a computer program, reading electronic mail, or using a spreadsheet. An important feature of HCI is that it is mostly a cognitive skill. It is engaged in mostly by people who already have a substantial experience with HCI and the kinds of tools and systems involved. At each sitting, the total task takes minutes to hours - 100 sec to 10^4 sec. Each total task is organized by the user into a hierarchy of tasks and subtasks. The lowest level in this hierarchy is at the level of the architecture: the processors of the MHP. A higher level in the hierarchy has already been introduced, the *unit task* (p. 1). These short segments of accomplishment can be held in mind, planned and executed, taking about 5-30 sec to perform (Card, et. al., 1983). Users accomplish tasks, learn new systems, and improve their skill, all within this framework. However, the behavior characterized by this framework does not include all modes of interaction with a computer. For instance, the initial period of becoming familiar with computers and learning to use a first computer system need not be organized this way. In addition, periods of open problem solving in which the system is unfamiliar or aberrant, or in which problem solving exists outside the interaction, may have a very different structure of behavior. Still, a large and important percentage of HCI can be cast in terms of a hierarchy and unit tasks.

Above the unit task level, the task-subtask hierarchy is homogeneous; the processes are goal setting, task definition and task execution, essentially sequential operations because they depend on each other. Large tasks are broken down into smaller subtasks until those subtasks are at the level of unit tasks (Figure 5-1).

The unit task is the lowest level task that can be planned. There are two limitations on planning. First, a lack of information stops a plan. For instance, when editing text from a marked-up manuscript, unit tasks are defined by the editorial markings. If the next marking is on the next page, the next unit task cannot be planned until the page is turned and the marking interpretted; this is a natural boundary for a unit task. Second, human short term memory is limited. Long, complex plans cannot be kept in short term memory, so planning at this level is limited to the execution of one unit task at a time.

At the next lowest level in the task hierarchy, unit tasks are composed of two qualitatively different

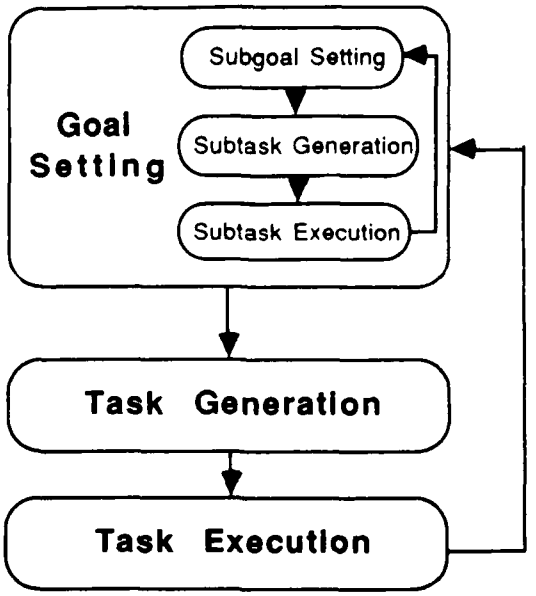


Figure 5-1: Processes involved in accomplishing a total task

types of behavior: deliberate actions and integrated skills, or *activities*. A deliberate action is a unique operation that the person composes and executes in response to the demands of a unit task. Each such action is serial, because it is deliberate, and the actions themselves are at the level of elementary MHP processes - keystrokes, comparisons, etc. Deliberate actions are at the level of the MHP architecture, the lowest level of the hierarchy.

Activities are skilled sequences of elementary processes that recur in many unit tasks and, within that framework, last from about 1 sec to about 10 sec at a time. For instance, transcription typing and reading are activities. The sequences of elementary operations of the perceptual, cognitive and motor processors that make up an activity are integrated, as they were in the METT. Activities are often performed sequentially within a unit task and their temporal boundaries are often easily identifiable when observing a user interacting with a computer system. Although several activities may be found together in a single unit task, they seem separable; reading is a distinct psychological process from typing, and has quite different properties as a skill. There seem to be a small number (tens, not hundreds) of such activities that show up at the interaction level of HCI and they each seem to have a small number of robust first-order effects that influence their execution. These properties of activities make them appropriate targets for engineering model development.

Activities can be identified by examining the behavioral loop of unit-task execution at the interaction level (Figure 5-2). For each unit task, the user must get information from the computer (or other local hard copy, like a marked-up manuscript), process that information to come up with a plan of action either by formulating a new plan or using one previously acquired (possibly modified), execute the planned actions and process the computer's response. Surrounding these processes are learning and errors, processes that occur at all stages of a unit task. Figure 5-3 displays the activites that will be described at these stages of unit task execution.

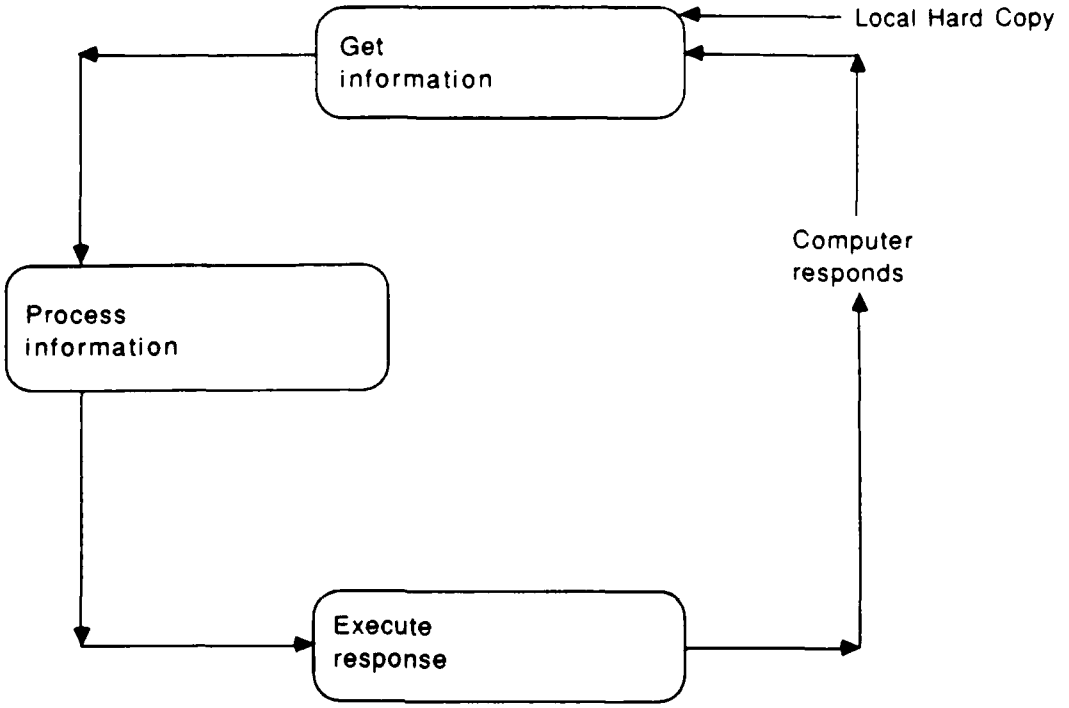


Figure 5-2: Unit Task Execution at the Interaction Level

5.1.1 Get Information

Two input channels commonly exist for getting information to the user of a computer system: visual and auditory. Visually, the user must be able to find information, the activity of *visual search*. *Reading* is probably the most common activity to get information in computer-to-human communication. If information is displayed graphically, *diagram interpretation* is also an activity.

Although not yet as well developed technologically, the auditory channel can be used for simple or complex communications, (e.g., a beep signals an illegal command, or voice mail, respectively). Simple communication is modelled directly by human characteristics inherent in the MHP (e.g., perceptual processor cycle time dictates the minimum duration of a beep) and is therefore not an activity, but a unique action. However, *listening*, the auditory analogue of reading, is an activity.

5.1.2 Process Information

Processing information seems to involve a host of different activities. In fact, these activities are well ordered in terms of how routine they are. For the least routine end, the user must engage in *problem solving* to discover an appropriate response. This is a mode of behavior that often moves outside the unit-task organization. Even if appropriate responses are clear (after problem solving or from previously stored experience), they must be organized into a sequence by a *planning* activity. There may be several appropriate plans (again, either stemming from the planning activity or coming from memory); one must be *selected*. Once a plan is selected (either through the selection activity or because it was stored in memory as the best thing to do), it must be *implemented*. To implement a plan, at each step in the plan, the situation is simply *mapped* onto the response. In this continuum of processing activities, the activities

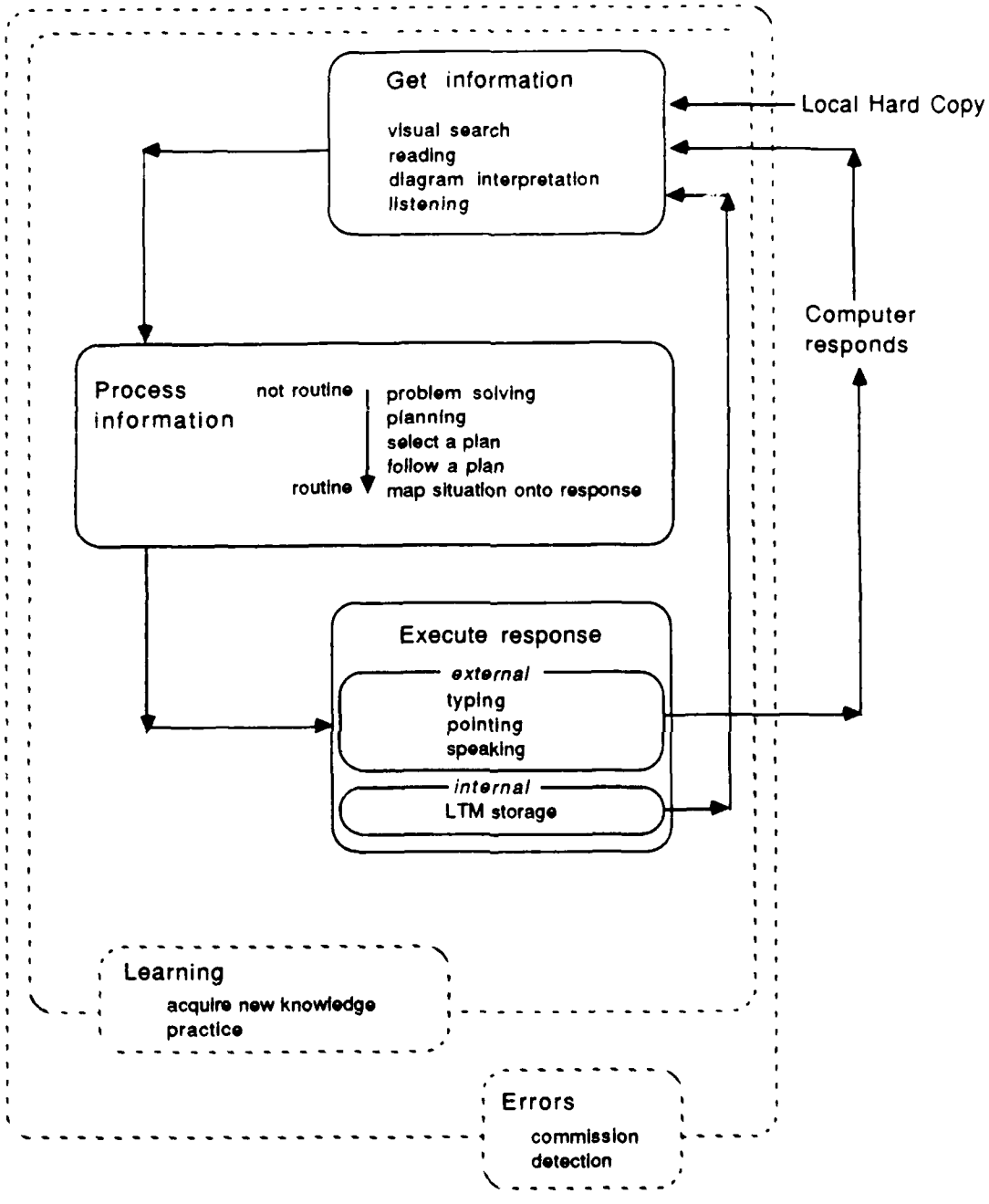


Figure 5-3: Activities Involved in an Interaction Level Unit Task Execution

necessary for less routine situations can be skipped if the appropriate responses have been learned through previous experience, i.e., if the task is routine at that level.

It seems to be primarily at the process-information stage that expert-novice differences appear. With experts, the information processing is toward the routine end of the continuum; with novices, it is more problematic. Experts seem to have acquired enough information about the computer system to have stored plans, and rules for selecting between them, for many situations encountered in the use of the

system. Novices, on the other hand, have to ponder the question of what to do in each new situation, forcing their behavior toward the problem-solving end of the information-processing continuum. The transition from novice to expert, then, might simply involve acquiring the appropriate information about the computer system and unit tasks (see 5.1.4, p. 91).

5.1.3 Execute Response

As with getting information, only a few modes of response are commonly used in human-to-computer communication. The most common form of human response is *typing*. *Pointing* is becoming a more available method of communication with many menu-based mouse-equipped computers. *Speech* is still uncommon, but is used with impoverished access terminals (e.g., the telephone). Other more exotic forms exist (e.g. eye-movements, gesture recognition), but only in research or specialized environments (e.g., severely handicapped users). Other physical responses also exist when communicating to a computer, e.g., pressing a mouse button. However, this response is an unique action, much as the perception of a beep is an unique action, and can be explained by the mechanisms of the MHP directly. Thus, only a few activities warrant engineering models at this stage of the interaction loop.

5.1.4 Learning

Learning is a process that surrounds every activity in unit task execution. Three types of learning can be identified: *acquiring knowledge about the computer system, practicing previously learned procedures*, and the *deliberate acquisition of new knowledge*. Acquiring knowledge about the system and practicing the execution of unit tasks occur continuously whenever the user is interacting with the system. Acquiring information about the system primarily involves acquiring plans for accomplishing certain goals so they can be used directly or modified when similar goals occur in later use of the system. Acquiring information about the structure of the task allows similarities in goals to be recognized as they are encountered, so the appropriate plans can be evoked. This is the process by which tasks become more routine and problematic stages in the information processing stage can be skipped.

Practice is the change in execution of an activity caused by repetition of that activity. The effects of practice occur without deliberate action and are ubiquitous in human behavior (Newell & Rosenbloom, 1981). All of the activities within unit task execution can improve with practice: visual search, reading, problem solving, typing - all of them. Acquiring system knowledge and practice are not activities in the same sense as the integrated skills like reading and typing, but engineering models of how they occur are necessary for a completely integrated theory of computer usage.

The deliberate acquisition of new knowledge can be thought of in terms of unit tasks. The deliberate act of acquiring new knowledge involves getting information, processing that information and executing a response. In this case, the response is an internal one involving the storage of the information in LTM. Elaboration of the information is done in the process-information stage and ranges from problematic to routine according to the familiarity of the learning situation. Thus, deliberate learning is simply another unit task, involving activities already described, and appears within the unit-task execution loop rather than surrounding it, as do the other two types of learning.

5.1.5 Errors

Three distinct processes are involved in errors: *commission, detection,* and *handling* them after detection. Each processor of the MHP is a source of some types of errors: the perceptual and cognitive processors could read a word incorrectly, the cognitive processor could set an incorrect intentional action, the motor processor could incorrectly execute a correctly intended action. Even though it is not an integrated skill, an engineering model of why and when errors occur is necessary for a complete theory of computer useage. Detection of errors seems to integrate perception, cognition and perhaps interruption of the motor system. Error handling, on the other hand, is composed of unit tasks (Card, Moran, & Newell, 1983, pp. 184-187). Once an error is committed and detected, a goal is set to fix the error and unit tasks are generated to accomplish that goal. The user must get information about the error and where it occurs in the total task, process that information to come up with a plan to correct it, then execute the responses necessary for that correction. As with any unit task, the processing of the information can be problematic or routine depending on the situation. As with deliberate learning of new knowledge, error handling is simply another unit task, within the unit-task execution loop, not surrounding it.

5.1.6 An Example of Activities in a Total Task

A examination of any total task accomplished with a computer shows the activities described above in abundance and also shows that they occupy the bulk of the time involved in accomplishing the task. For instance, consider the tasks given by Olson and Nilsen (1987) in their study of experts using spreadsheets. The entry task included unit tasks such as "setting column widths, formatting items in columns, entering numbers and formulas, saving the spreadsheet, and printing it." The modification tasks included unit tasks of "accessing the stored spreadsheet, adding another column containing a formula, changing the column widths, inserting rows and columns, altering the formatting and location of portions of the spreadsheet, changing a value and reporting the related changes, printing part of the spreadsheet, and saving the modified version."

Walking around the unit task execution loop with a few of these unit tasks, within a specific spreadsheet software package (Excel for the Apple Macintosh[10]) will demonstrate the coverage of the activities identified. Setting column widths requires the user to find the column to be set on the hard copy (the subjects were told to reproduce a hard copy spreadsheet on their screens), and its counterpart on the screen through *visual search*. She must then recall the procedure for setting column widths, much like recalling the command abbreviations in Chapter 3 of this dissertation, an activity at the most routine end of the process information stage (*map situation onto response*). She then must execute this procedure by *pointing* to the column with the mouse, see the cursor change to the appropriate icon *diagram interpretation*, and drag the cursor (*pointing* again) so the column is the appropriate width.

A second brief example involves entering a formula into a cell, a task Olson and Nilsen call "the seat of cognition in spreadsheet tasks". To accomplish this task in Excel, the user must first *read* the hard copy so as to understand the desired effect of the formula. Depending on the familiarity of the specific effect desired, she might have to go into a *problem solving* activity to derive the necessary formula, or simply remember it from previous use. She would then have to form a plan to implement the formula, through deliberate *planning* or recall of appropriate plans and *selection* between them. Execution of this plan first involves indicating the cell into which the formula should be entered by *pointing* to it and clicking the

[10]Microsoft Corporation holds the copywrite for Excel, Apple and Macintosh are trademarks of Apple Computers, Inc.

mouse button. Then the formula must be indicated by *pointing* to the formula menu, pressing the mouse button, dragging to the paste-formula item, and clicking on it. A dialogue box appears and the user must *visually search* it for the desired formula, which must then be pointed to and double-clicked. The appropriate arguments for the formula must then be selected by *visually searching* for them, and indicating them with *pointing* and mouse button clicks. Again, all the activities necessary for accomplishing this unit task have been identified in the task analysis given above.

Now that a list has been compiled of activities at the interaction level of the human-computer interface, the current state of model development can be reviewed and this dissertation positioned within it.

5.2 Engineering Model Development in HCI

Two types of theoretical development contribute to the state of engineering models in HCI: development specifically aimed at understanding HCI activities, and development of cognitive theories outside of HCI that relate to activities also found within HCI. The first type of theories often contribute engineering models that satisfy the criteria of making approximate, quantitative, zero-parameter predictions of human behavior and ultimately being useful in computer system design. The second type of theories do not usually have such pragmatic goals and require modification to be engineering models. This section will survey proposed models of both types that cover the activities at the interaction level. It is not the purpose of this section to present an exhaustive review of the theoretical literature, but to provide an overview of the state of the field sufficient to locate the contributions of this thesis. The models surveyed will not be judged as to whether they are already true engineering models, or how much modification is necessary to make them useful engineering models. Figure 5-4 indicates the models that cover the various activities found in HCI (excluding the work of this thesis).

5.2.1 Existing HCI Models

Existing models in HCI concentrate on explaining behavior in one or two activities, and make simplifying assumptions about the other activities necessary to accomplish a task. For instance, the METT explains some of the details of typing, but makes a simplifying assumption that reading of different words, syllables, and letters takes a constant amount of time. Thus, each model will be discussed in terms of that activity it primarily strives to explain in detail.

Several models of visual search exist within the HCI task domain (e.g., Card, 1983, MacGregor, Lee, and Lam, 1986). Card (1983) proposes an unsystematic search model relating the probability of finding a known menu item to the elapsed time of the search and a parameter, *m*, that reflects the arrangement of the items, practice, area to be searched, and visual conspicuity of the target. MacGregor, Lee and Lam (1986) propose a probablistic linear search model of videotext menu usage. This model makes several predictions about the effects of menu size on search strategy (and the search time patterns different strategies imply) and error commission. Both of these models are candidates for engineering models of visual search,[11] for their parameters could be estimated through a series of laboratory experiments to allow zero-parameter predictions to be made.

---

[11]There is some controversy in the literature as to whether current data differentiates between these models (MacGregor & Lee, 1987).

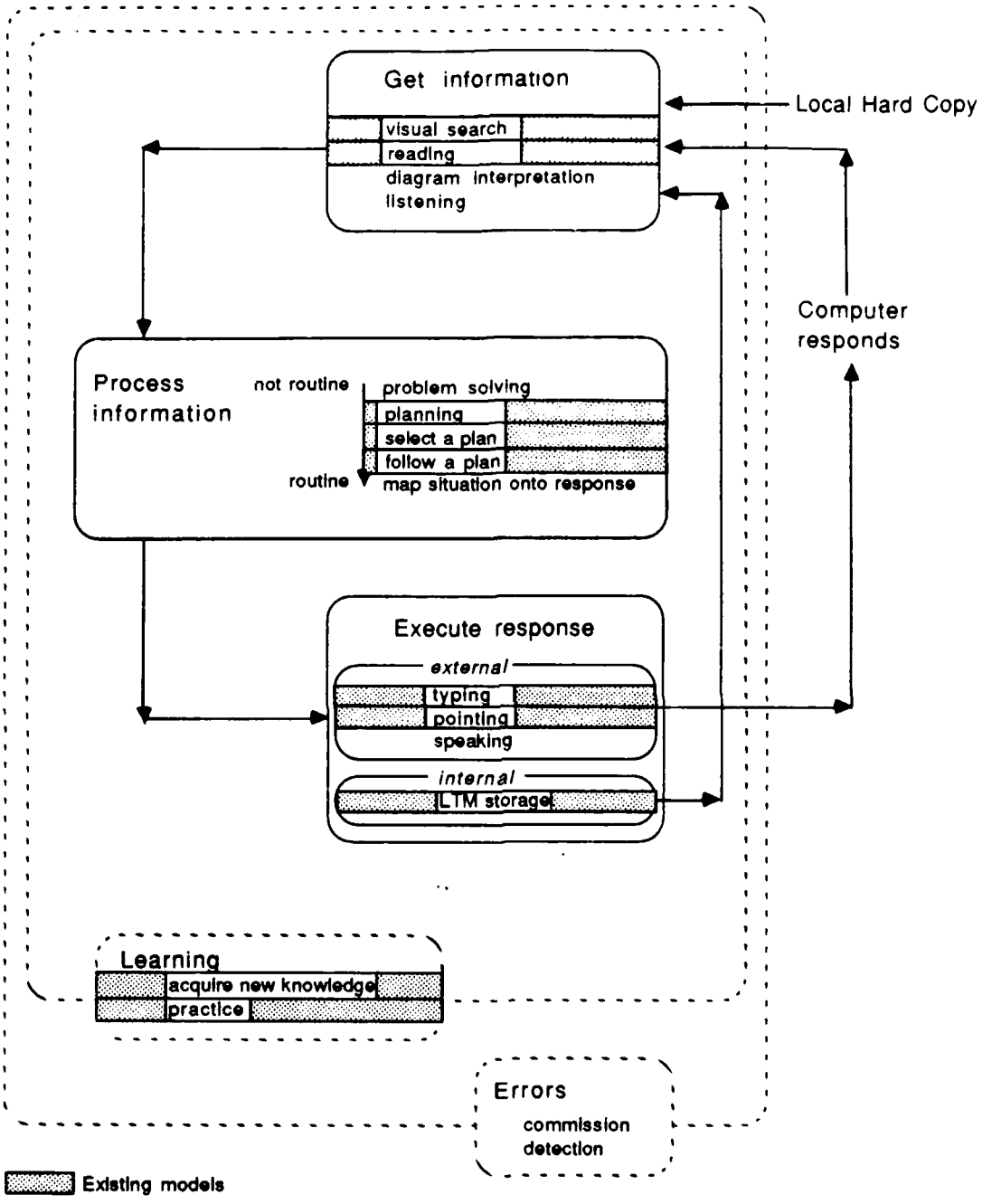


Figure 5-4: Existing Models of Activities in HCI

The GOMS model of command language use (Card, et. al., 1983), reviewed in Chapter 2 of this dissertation, predicts the user's sequence of actions while performing a routine total task. In the version presented here, these actions are on the level of the stages in Figures 5-2, 5-3, and 5-4: get-information, where the user looks at the marked-up manuscript or turns a page, and execute-response, where the user types a specific command. The major activities modelled by GOMS for command language use are *planning, selecting a plan,* and *following a plan.* This model is a model of expert behavior, so planning is

not problematic. It involves retrieving previously used plans from LTM and selecting between them to create a sequence of operations that minimize the time to complete a total task.

The Keystroke-Level Model (KLM), a version of GOMS at a smaller grain size, is a model of the execute-response stage of a unit task. It contains a simple model of *typing*: each keystroke takes an average amount of time and a mental operator precedes a burst of typing. It also provides a model of *pointing* that uses an average time predicted by a Fitts' Law analysis.

Polson and Kieras (Polson & Kieras, 1985) propose a model of users' knowledge of a computer system as a production system. These rules of "how-to-do-it" knowledge provide a means of predicting learning time for a set of computer commands (Polson & Kieras, 1985), transfer of learning between different commands (Polson, Muncher & Engelbeck, 1986), and performance time for accomplishing a task (Polson & Kieras, 1985). The time to learn each production (*declarative knowledge acquisition*) is estimated to be a constant, and thus the time to learn a command is the sum of the time to learn each production necessary to recognize the need for and execute that command. A common elements theory of transfer of learning is assumed, where the elements are the productions associated with a command. The time to learn an additional command is therefore only the time to learn the additional productions necessary for that command; the savings due to transfer is equal to the sum of the times associated with the productions in common with previously learned commands. It is also assumed that each production takes a constant time to execute, so performance time for a task can be predicted by the number of production firing needed to accomplish that task.

A causal model of *learning by example* was proposed by Lewis (1986) and extended by Casner and Lewis (1987); it explains the acquisition of a mental model about a computer system. Given a series of user actions and computer responses to those actions, the model predicts what roles an observer will assign to each action. This model could be used to predict the *commission of errors*, perhaps both type and quantity, made by novices as they build up their knowledge of a system.

An engineering model of *practice* exists in the MHP: principle 6, the power law of practice (Newell & Rosenbloom, 1981,p. 10 in this dissertation). This model is expressed as a mathematical formula relating the time to perform the *n*th trial of a task to *n*, the number of trials performed. Although different mechanisms might cause this speed-up (e.g., speed-up of an operator as the motor operator is assumed to speed-up in the METT, or a shift in strategy to one that is more efficient), this relationship is a good approximation to the overt behavior no matter what the mechanism. The power law relationship seems to be ubiquitous, giving the model extremely broad coverage for HCI tasks.

5.2.2 Other Cognitive Models

Regression-based models of reading have the flavor of engineering models (see Britton and Black, 1985, for several different reading models). One example is the model proposed by Just and Carpenter (1987). This model is embodied in a computer simulation called READER that reads and interprets text, producing a prediction of the time course of the reading process and of the content of the information a human reader would abstract from the text. Computer simulations of human processes are an attractive form for engineering models, because much of the psychology of the model is built into the computer program and is transparent to a user of the simulation. In this case, a user of this theory need not learn all about how, when, and where eye-fixations occur; the task is reduced to how to represent the text, and the background knowledge necessary to understand it, in a form READER can use.

Current models of listening seem not to be developed in a form amenable to engineering use. Jusczyk (1986) states

> It is fair to say that there are no complete models of on-line sentence processing. What models have been proposed to date tend to focus extensively on one aspect of speech processing, to describe it in great detail, and to leave unspecified the kind of processing that goes on at other levels...the basic approach in the past has been to try to break the process of on-line comprehension down into a series of subcomponents and to attempt an accurate description of each, with the hope that at some future point the various components could be assimilated into some overall model." (p. 27-43)

Jusczyk goes on to discuss several of these component models (Morton's logogen model, 1969, Forster and Olbrei's autonomous search model, 1973, Marslen-Wilson's interactive theory, 1975, HEARSAY by Reddy and his colleagues, 1973, Lowerre's HARPY, 1976, and Klatt's model of speech recognition, 1979), many of which could be the starting point for the integration and simplification necessary for an engineering model of listening.

A model of problem solving is inherent in the MHP, as evidenced by principle of operation number 9, the Problem Space Principle (page 10). This principle is based on the work of Newell and Simon (1972) laying out a theory of human problem solving based on four propositions:

1. A few, and only a few, gross characteristics of the human [information processing system] IPS are invariant over task and problem solvers.[12]
2. These characteristics are sufficient to determine that a task environment is represented (in the IPS) as a problem space, and that problem solving takes place in a problem space.
3. The structure of the task environment determines the possible structures of the problem space.
4. The structure of the problem space determies the possible programs that can be used for problem solving. (pp. 788-789)

This general theory has spawned many concrete models. A recent unified theory of cognition, Soar, characterizes all behavior as search through a problem space, and has been demonstrated to model a wide range of human behavior from the immediate behavior tasks discussed in Chapter 3 of this dissertation, to complex algorithms design, to the acquisition of reasoning skill in a classic developmental task (Laird, Newell, & Rosenbloom, 1987; Newell, 1987).

As with models of listening, models of speaking are not yet useful for engineering purposes. Ellis' volumes, *Progress in the Psychology of Language* (1985), present several models of language production (e.g., Stemberger's interactive activation model, Robinson and Moulton's model of language cognition, Barnard's interactive cognitive subsystems). These models begin to integrate many language production tasks, and language understanding as well, but they do not make quantitative, zero-parameter predictions relevant to HCI tasks. Again, as with models of listening, these could be starting points for engineering model development.

A model of LTM storage and retrieval (Chase and Ericsson, 1981; Staszewski, 1987) has the predictive flavor needed for an engineering model of the process. The task modeled is the expert recall of random digits. The basic premises are that (1) information can be chunked into a pre-existing retrieval structure that can aid in organizing information and in the retrieval of that information at a later time, and that (2) a person can be trained to use a particular retrieval structure. Knowing the retrieval structure allows prediction of the time to retrieve any particular piece of information. It also allows prediction of some of the types of errors that might be made in retrieval. The model is tailored to a particular digit-chunking scheme

[12] In the MHP, these invariants are the processors, memories and principles of operation.

used by a particular expert for an artificial task. Although, as it stands, this model has little application in HCI tasks, it serves as an existence proof that these premises have some validity. It is also easy to envision application of this model to HCI tasks: proposing a retrieval structure for computer commands that can be analytically evaluated, teaching users an optimum retrieval structure, predicting performance based on knowledge of the retrieval structure that is taught, etc.

5.3 The Contribution of this Dissertation to HCI

The immediate behavior work presented in Chapter 3 of this dissertation contributes to a set of HCI engineering models in two ways. First, it demonstrates that SRC effects are present in HCI tasks. These effects are not surprising, because the most routine activity in the information-processing stage of the unit task execution loop is mapping the situation onto a response, exactly the task modelled by the GOMS model of immediate behavior. However, not much attention has been directed toward SRC effects in the HCI domain, despite the promenance of SRC in human factors generally, and this demonstration contributes by pointing out the existence of those effects.

Second, the GOMS model of immediate behavior has been shown to make quantitative, zero-parameter predictions to within 20% of observed performance. This model is therefore an engineering model of reaction time tasks, the routine mapping of a situation onto a response, and fills a theory gap in the activities list presented in the task analysis of HCI (Figure 5-5).

In addition, this immediate-behavior model, combined with a chunking theory of learning and practice has been demonstrated to predict the learning curve as subjects move from novice to expert in a perceptual-motor RT task (Rosenbloom, 1983). The model's analyses of tasks within the domain of HCI can also be expected to display this extension from predictions of performance to predictions of the effects of practice.

The METT makes predictions about the performance time of experts on transcription typing tasks. Although, the METT has been demonstrated as a model of transcription typing, it may be extended to other similar tasks like original typing (typing material composed by the typist rather than copying written text), typing from dictation, and other keying tasks. For instance, in original typing, the perceptual operators would be replaced by cognitive operators that generate the material to be typed. Although this is a genuine extension of the model, and needs empirical confirmation, the METT itself gives strong grounds for expecting the predictions to be confirmed. The METT could then be combined with many of the other models presented in this analysis. For instance, if Ericsson and Chase's mneumonist were to have typed his list of random numbers (and he had been an expert typist), this model could be used combined with their model of LTM retrieval to predict the interkeystroke intervals.

Typing was also modelled by the Card, et. al.'s KLM (1983). Multiple coverage of an activity does not detract from the contribution of the METT, because of the approximate nature of engineering models. If the analysis of a typing task requires only a rough approximation of total performance time, then the fixed duration per keystroke of the KLM suffices. However, if the task requires more fine-grained analysis, as did the experimental tasks presented by Salthouse (1986), then the KLM is not adequate and the METT steps in. As one would expect, the METT reduces to the KLM by taking the average interkeystroke time as the duration of all keystrokes. The KLM uses 0.12 sec for a 90 wpm typist and 0.2 sec for a 55 wpm typist; the METT reduces to 0.13 sec for a 90 gwpm typist and 0.2 sec for a 60 gwpm typist. At the level of analysis of the KLM, these slight differences are negligible.

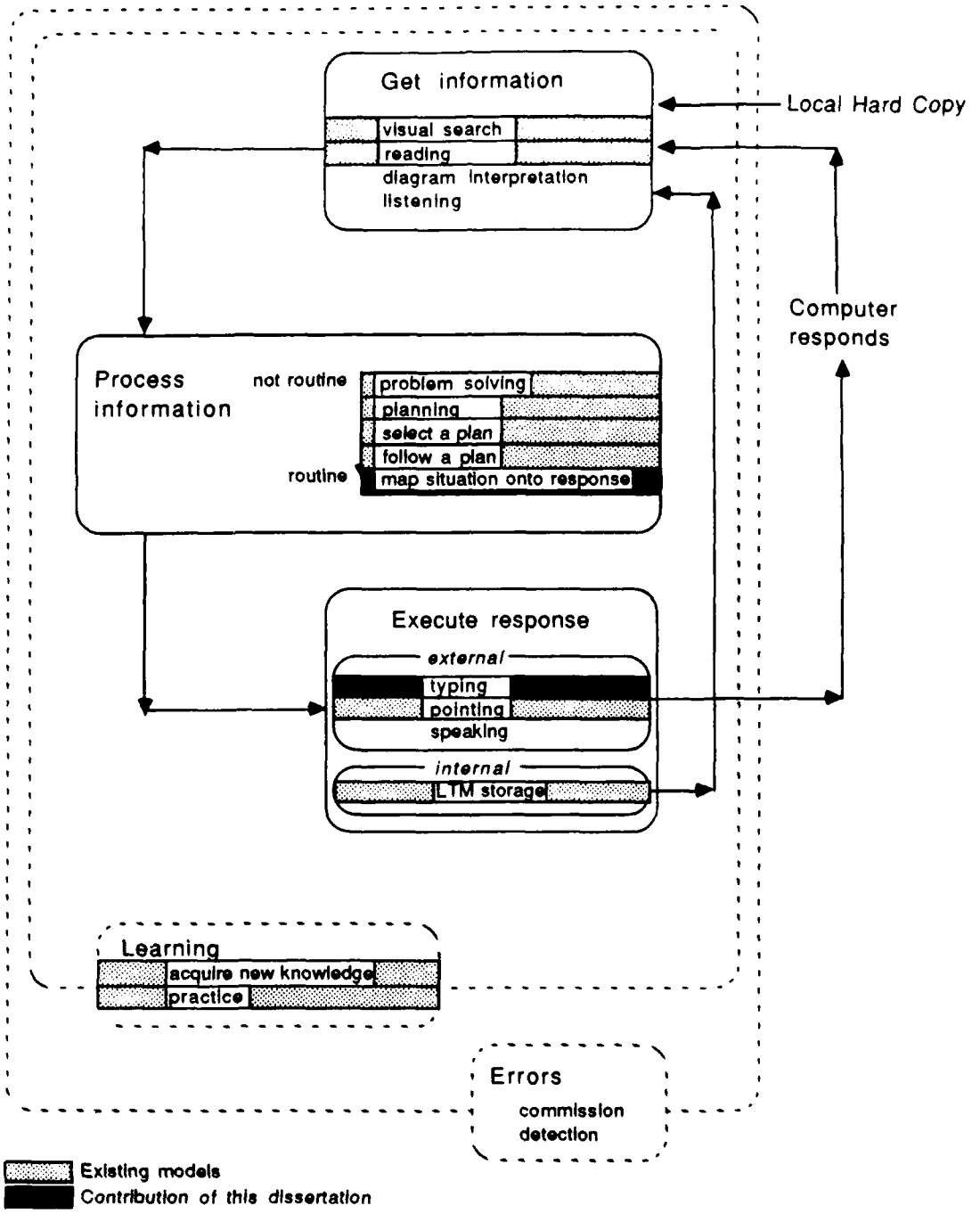


Figure 5-5: The HCI activities modelled by this dissertation, shown in black

Thus, this dissertation contributes two new engineering models of activities involved in the unit-task execution loop of HCI. Since the task analysis of the unit-task execution loop includes only 18 activities and related actions (e.g., commision of errors), this dissertation contributes about 10% of the models necessary for a complete theory of HCI at the interaction level and only leave about 25% to go.

The GOMS model of immediate behavior and the METT contribute to HCI in ways other than simply

filling in gaps in the collection of activity models. The success of these models highlights the way approximation operates in engineering models. GOMS had been used at several levels of operator durations previous to this work (Card, et. al., 1983), but never at so low a level. The ability for GOMS to extend downward toward the MHP itself supports the contention that the MHP is a good basis for building engineering models.

In addition, the research presented here suggests some changes to the MHP. Most notably, the empirical results of the command abbreviation experiments indicate that the cognitive processor cycle time of the MHP can be narrowed from the 25 msec to 170 msec range given by Card, Moran and Newell to about 50 msec (e.g., 40-60 msec). An inspection of the tasks that produced the broader range could be then be made using the GOMS model of immediate behavior. Similarly, an inspection of the MHP motor processor cycle time is indicated by the success of the METT motor operator that varies with experience. This dissertation, therefore, points clearly toward a technique for producing a more precise definition of the MHP.

Finally, another feature of the models presented in this dissertation is that they are truly integrated. They spring from the same base, the MHP. They use the same parameter estimates. It has even been demonstrated that they can be woven together to predict behavior in a dual-task situation that includes both domains (see typing Phenomenon 12, section 4.2.1.12, page 63). These models are also GOMS models with the same hierarchical structure displayed in several other GOMS analyses. For instance, it is easy to see how they would fit onto the low end of the Card, Moran and Newell text-editing analyses; after a plan had been selected, command names would be recalled with the model of immediate behavior and then typed out with the METT. Unfortunately, most of the other models mentioned in this quick look at the HCI field are not as integrated as the two models present here. However, such integration is a necessity. Only an engineering model that encompasses all activities at the interaction level can allow the analyses of total tasks to become standard procedure in the design of human/computer interfaces.

6. Conclusion: HCI and Science, Too

The goals of this dissertation were to extend, consolidate, and make concrete engineering models based on the MHP. In pursuit of these goals, the GOMS model was extended to a lower level of detail, with operators on the order of a single cycle time of the MHP's cognitive processor. The extended GOMS model was then successfully applied in the domains of immediate behavior and transcription typing. Before this dissertation, the HCI activities of mapping a situation onto a response (the activity modelled by the immediate behavior work) and typing had not been explained in an engineering model at this level. In addition, the use of one basic procedural structure, GOMS, within a unified theory of human behavior, the MHP, to explain such diverse tasks demonstrates a consolidation not previously realized. The application of the models is made concrete through examples of their use. The version of GOMS used to analyze typing tasks is especially concrete, with six specific assumptions constraining the analysis of a task. The goals of the dissertation have been accomplished.

There remains one bit of unfinished business: What has this work contributed to the more basic enterprise of cognitive psychology?

Engineering models satisfy criteria that distinguish them from traditional cognitive models: zero-parameter predictions, useability by practitioners, coverage, and approximation. These criteria only serve to restrict the form of cognitive model that can be considered an engineering model. They do not define an entirely different breed of model. Thus, engineering models are simply points in a continuous space of cognitive models. The dimensions of this space are exactly the criteria of an engineering model: the number of free parameters needed to explain data, the ease of use, the breadth of coverage, and the level of approximation of the model. For instance, as mentioned when discussing READER (Section 5.2.2, page 95), a cognitive model embodied in a computer simulation may be detailed and conceptually difficult, and yet accessible to practitioners because the complexity of the psychology inherent in the model is hidden from the user behind a simple interface. Along the dimension of approximation, the METT was presented as differentiating between hands but not between fingers on a hand. Choosing a high level of approximation, not differentiating between hands and eliminating the parallel operation of the three MHP processors, the METT reduces to the KLM with a single parameter for the duration of each keystroke. Likewise, an extension of the METT that uses Fitts' Law to predict the movement of individual fingers further down the dimension of approximation explains as much of a very detailed phenomenon as the more traditional Rumelhart and Norman model (1982) (see Section 4.2.1.11, page 60).

Given this demonstration of the continuum of cognitive models, this dissertation contributes to cognitive psychology in three ways. First, and most narrowly, it presents predictive models of SRC and transcription typing. These models not only satisfy the criteria for engineering models, they stand as basic cognitive theory. As far as we know, there is no other theory of typing that predicts, quantitatively, as many of the Salthouse 29 as the METT predicts. Certainly, there is no model of typing that also predicts stimulus-response compatibility.

Second, since these models share a common structure with each other and with the GOMS model that has been applied to the use of text-editors, the design of IC circuits (Card, et. al., 1983), learning text-editor commands (Polson & Kieras, 1985), and the use of spreadsheets (Olson & Nilsen, in press), this dissertation demonstrates the validity and power of using a unified theory of human performance to develop cognitive models. Another example of this power is the examination of the Rumelhart and Norman task where the first step is made in bringing together Fitts' Law and transcription typing. This is a

very small and overly simple attempt, and it needs much more careful elaboration and extension. Yet we know of no other demonstration that shows that Fitts' Law applies to typing, just as it does to a wide range of simple motor behavior. Such are the fruits of an unified theoretical framework.

Third, the demonstration of the continuous space of cognitive models itself is a contribution, dispelling an impression that developing applied, engineering models is an occupation isolated from the main stream of cognitive psychology.

There are three paths along which this work should be pushed. Of interest to HCI and human factors researchers, the activities of the HCI unit-task execution loop must be filled in with engineering models; the goals of extension, consolidation, and concreteness are not complete until all these activities are covered. Of interest to cognitive psychologists, the use of unified cognitive architectures should expand to many other psychological domains. Of interest to computer designers, the usefulness of psychological engineering models to the design of computers must be demonstrated in practice far beyond the few examples that currently exist.

References

Allen, R. B., & Scerbo, M. W. (1983). Details of command-language keystrokes. *ACM Transactions on Office Information Systems, 1*(2), 159-178.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, Mass: Harvard University Press.

Anderson, J. R. & Reiser, B. J. (1985, April). The lisp tutor. *Byte*, pp. 159-160, 162, 164, 166, 168, 170, 172, 174-175.

Bailey, R. W. (1982). *Human performance engineering: A guide for system designers.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Baron, J. (1978). Intelligence and general strategies. In G. Underwood (Ed.), *Strategies of information processing* . London: Academic Press.

Beakley, G. C., Evans, D. L., & Keats, J. B. (1986). *Engineering: An introduction to a creative profession.* New York: Macmillan Publishing Co.

Borenstein, N. S. (1985). The evaluation of text editors: A critical review of the Roberts and Moran methodology based on new experiments. In L. Borman and B. Curtis (Eds.), *CHI'85 Human Factors in Computing Systems (San Francisco, April 14-18).* New York: ACM, 99-106.

Brebner, J. (1973). S-R compatibility and changes in RT with practice. *Acta Psychologica, 37*, 93-106.

Britton, B. K., & Black, J. B. (1985). *Understanding expository text: A theoretical and practical handbook for analyzing explanatroy text.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Brown, J., and Huda, M. . (1961). Response latencies produced by massed and spaced learning of a paired-associate list. *Journal of Experimental Psychology, 61*, 360-364.

Butsch, R. L. C. (1932). Eye movements and the eye-hand span in typewriting. *Journal of Educational Psychology, 23*, 104-121.

Card, S. K. (1983). Visual search of computer command menus. In H. Bouma and D. Bouwhuis (Ed.), *Attention and performance X* . Hillsdale, NJ: Lawrence Erlbaum, Associates.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human computer interaction.* Hillsdale, NJ: Lawrence Erlbaum, Associates.

Casner, S. & Lewis, C. (1987). Learning about hidden events in system interactions. In J. M. Carroll & P. P. Tanner (Eds.), *CHI+GI 1987 (Toronto, April 5-9).* New York: ACM, 197-203.

Chase, W. G., and Ericsson, K. A. (1981). Skilled memory. In J. R. Anderson (Ed.), *Cognitive skills and their acquistion* . Hillsdale, NJ: Lawrence Erlbaum, Associates.

Cheatham, P. G., & White, C. T. (1954). Temporal numerosity: III. Auditory perception of number. *Journal of Experimental Psychology, 47*, 425-428.

Deininger, R. L. & Fitts, P. M. (1955). Stimulus-response compatibility, information theory, and perceptual-motor performance. In H. Quastler (Ed.), *Information theory in psychology*. Glencoe, Illinois: Free Press.

Duncan, J. (1977). Response selection rules in spatial choice reaction tasks. In S. Dornic (Ed.), *Attention and performance VI*. Hillsadle, NJ: Lawrence Erlbaum Associates.

Andrew W. Ellis, ed. (1985). *Progress in the psychology of language*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Filbey, R. A. & Gazzaniga, M. S. (1969). Splitting the normal brain with reaction time. *Psychonomic Science, 17*, 335-336.

Fitts, P. M., & Seeger, C. M. (1953). S-R compatibility: Spatial characteristics of stimulus and response codes. *Journal of Experimental Psychology, 46*, 199-210.

Forster, K. I., & Olbrei, I. (1973). Semantic heuristics and syntactic analysis. *Cognition, 2*, 319-347.

Gentner, D. R. (1983). The acquisition of typewriting skill. *Acta Psychologica, 54*, 233-248.

Henderson, L. (1982). *Orthography and word recognition in reading*. London: Academic Press.

Hershman, R. L., & Hillix, W. A. (1965). Data processing in typing: Typing as a function of kind of material and amount exposed. *Human Factors, 7*, 483-292.

Hilgard, E. R. (1951). Methods and procedures in the study of learning. In S. S. Stevens (Ed.), *Handbook of experimental psychology*. New York: John Wiley and Sons, Inc.

Jensen, A. R. & Rohwer, W. D. (1963). Verbal mediation in paired-associate and serial learning. *Journal of Verbal Learning and Verbal Behavior, 1*, 346-352.

John, B. E., & Newell, A. (1987). Predicting the time to recall computer command abbreviations. In J. M. Carroll & P. P. Tanner (Eds.), *CHI+GI 1987 (Toronto, April 5-9)*. New York: ACM, 33-40.

John, B. E., Rosenbloom, P. S., & Newell, A. (1985). A theory of stimulus-response compatibility applied to human-computer interaction. In L. Borman and B. Curtis (Eds.), *CHI'85 Human Factors in Computing Systems (San Fransisco, April 14-18)*. New York: ACM, 213-219.

Juscryk, P. (1986). Speech perception. In K. R. Boff, L. Kaufman, and J. P. Thomas (Ed.), *Handbook of perception and performance* (pp. 27-1 - 27-57). New York: John Wiley and Sons.

Just, M. A. & Carpenter, P. A. (1987). *The psychology of reading and language comprehension*. Boston: Allyn and Bacon, Inc.

Kay, D. S. & Black, J. B. (1985). *Knowledge transformations during the acquisition of computer expertise* (Tech. Rep. CCT Report 85-1). Department of Communications, Computing, and Technology, Teachers College, Columbia University.

Kieras, D. E. & Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. *Journal of Memory and Learning, 25*, 507-524.

Kieras, D. E. & Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22*, 365-394.

Klatt, D. H. (1979). Speech perception: A model of acoustic-phonetic analysis and lexical access. *Journal of Phonetics, 7*, 279-312.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*(1), 1-64.

Landauer, T. K., Dumais, S. T., Gomez, L. M., & Furnas, G. W. (November 1982). Human factors in data access. *Bell System Technical Journal, 61*, 2487-2509.

Larochelle, S. (1983). A comparison of skilled and novice performance in discontinuous typing. In W. E. Cooper (Ed.), *Cognitive aspects of skilled typewriting* (pp. 67-94). New York: Springer-Verlag.

Lewis, C. (1986). A model of mental model construction. In M. Mantei and P. Orbeton (Eds.), *CHI'86 Human Factors in Computing Systems (Boston, April 13-17)*. New York: ACM, 306-313.

Logan, G. D. (1982). On the ability to inhibit complex movements: A stop-signal study of typewriting. *Journal of Experimental Psychology: Human Perception and Performance, 8*(6), 778-792.

Logan, G. D. (1983). Time, information, and the various spans in typewriting. In W. E. Cooper (Ed.), *Cognitive aspects of skilled typewriting* (pp. 197-224). New York: Springer-Verlag.

Lowerre, B. T. (1983). *The Harpy speech recognition system.* Doctoral dissertation, Carnegie-Mellon University.

Luce, R. D. (1986). *Oxford Psychology Series.* Vol. 8: *Response times: Their role in inferring elementary mental organization.* New York: Oxford University Press.

MacGregor, J. & Lee, E. (May 1987). Menu search: random or systematic? *International Journal of Man-Machine Studies, 26*(5), 627-631.

MacGregor, J., Lee, E. & Lam, N. (August 1986). Optimizing the structure of database menu indexes: A decision model of menu search. *Human Factors, 28*(4), 387-400.

Marslen-Wilson, W. (1975). Sentence processing as an interactive process. *Science, 189*, 226-228.

Millward, R. (1964). Latency in a modified paired-associate learning experiment. *Journal of Verbal Learning and Verbal Behavior, 3*, 309-316.

Montague, W. E., Adams, J. A, & Kiess, H. O. (1966). Forgetting and natural language mediation. *Journal of Experimental Psychology, 72*, 829-833.

Morin, R. E. & Forrin, B. (1962). Mixing two types of S-R associations in a choice reaction time task. *Journal of Experimental Psychology, 64*, 137-141.

Morin, R. E. & Grant D. A. (1955). Learning and performance on a key-pressing task as function of the degree of spatial stimulus-response correspondence. *Journal of Experimental Psychology, 49*(1), 39-47.

Morton, J. (1969). Interaction of information in word recognition. *Pychological Review, 76*, 165-178.

Newell, A. (1973). Production systems: Models of control structures. In W. G. Chase (Ed.), *Visual information processing* . New York: Academic Press.

Newell, A. (1987). Unified Theories of Cognition. The William James Lectures. Harvard University, Spring 1987. (Available in videocassette, Psychology Department, Harvard).

Newell, A. & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* . Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A & Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Olson, J. R. & Nilsen, E. (in press). Analysis of the cognition involved in spreadsheet software interaction. *Human Computer Interaction.*

Ostry, D. J. (1983). Determinants of interkey times in typing. In W. E. Cooper (Ed.), *Cognitive aspects of skilled typewriting* (pp. 225-246). New York: Springer-Verlag.

Polson, P. G. & Kieras, D. E. (1985). A quantitative model of learning and performance of text editing knowledge. In L. Borman and B. Curtis (Eds.), *CHI'85 Human Factors in Computing Systems (San Francisco, April 14-18).* New York: ACM, 207-212.

Polson, P. G., Muncher, E., & Engelbeck, G. (1986). A test of a common elements theory of transfer. In M. Mantei and P. Orbeton (Eds.), *CHI'86 Human Factors in Computing Systems (Boston, April 13-17).* New York: ACM, 78-83.

Prytulak, L. S. (1971). Natural language mediation. *Cognitive Psychology, 2*, 1-56.

Rabbit, P. (1978). Detection of errors by skilled typists. *Ergonomics, 21*(11), 945-958.

Reddy, D. R., Erdman, L. D., Fennell, R. D., & Neely, R. B. (1973). The Hearsay speech understanding system: An example of a recognition process. In *Proceedings of the International Joint Conference on Artificial Intelligence.* Stanford, Calif.: AAAI, 185-194.

Roberts, T. L. (1979). *Evaluation of Computer text editors.* Doctoral dissertation, Stanford University.

Rosenbloom, P. S. (August 1983). *The chunking of goal hierarchies: A model of practice and stimulus-response compatibility.* Doctoral dissertation, Carnegie-Mellon University.

Rothkopf, E. Z. (1980). Copying span as a measure of the informatio burden in written language. *Journal of Verbal Learning and Verbal Behavior, 19*, 562-572.

Rumelhart, D. E., & Norman, D. A. (1982). Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science, 6*, 1-36.

Runquist, W. N. & Farley, F. H. (1964). The use of mediators in the learning of verbal paired associates. *Journal of Verbal Learning and Verbal Behavior, 3*, 280-285.

Salthouse, T. A. (1984). Effects of age and skill in typing. *Journal of Experimental Psychology: General, 113*(3), 345-371.

Salthouse, T. A. (1984). The skill of typing. *Scientific American, 250*, 128-135.

Salthouse, T. A. (1985). Anticipatory processes in transcription typing. *Journal of Applied Psychology, 70*, 264-271.

Salthouse, T. A. (1986). Perceptual, cognitive, and motoric aspects of transcription typing. *Pychological Bulletin, 99*(3), 303-319.

Salthouse, T. A. (1988). Private communication.

Salthouse, T. A. & Saults, J. S. (1987). Multiple spans in transcription typing. *Journal of Applied Psychology, 72*(2), 187-196.

Schweiker, H. & Muthig, K. P. (in press). Solving interpolation problems with LOGO and BOXER. In M. J. Tauber (Ed.), *Visual aids in programming* . .

Shaffe·, L. H. (1973). Latency mechanisms in transcription. In S. Kornblum (Ed.), *Attention and performance, IV* (pp. 435-446). New York: Academic Press.

Shaffer, L. H., & French, A. (1971). Coding factors in transcription. *Quarterly Journal of Experimental Psychology, 23*, 268-274.

Shaffer, L. H., & Hardwick, J. (1970). The basis of transcription skill. *Journal of Experimental Psychology, 84*, 424-440.

Shepard, Roger N. (1961). Role of generalization in stimulus-response compatibility. *Perceptual and Motor Skills, 13*, 59-62.

Singley, M. K. & Anderson, J. R. (in press). A keystroke analysis of learning and transfer in text-editing. *Human Computer Interaction.*

Smith, G. A. (1977). Studies of compatibility and a new model of choice reaction times. In S. Dornic (Ed.), *Attention and performance VI* . Hillsadle, NJ: Lawrence Erlbaum Associates.

Staszewski, J. (1987). *The psychological reality of retrieval structures: A theoretical and empirical investigation of expert knowledge.* Doctoral dissertation, Cornell University.

Stemberg, S., Knoll, R. L., & Wright, C. E. (1978). Experiments on temporal aspects of keyboard entry. In J. P. Duncanson (Ed.), *Getting it together: Research and applications in human factors* (pp. 28-50). Santa Monica, CA: Human Factors Society.

(1979). *Webster's New Collegiate Dictionary*. Springfield, Mass.: G. & C. Merriam Company.

West, L. J. and Sabban, Y. (1982). Hierarchy of stroking habits at the typewriter. *Journal of Applied Psychology, 67*, 370-376.

Wickens, C. D., Vidulich, M., & Sandry-Garza, D. (October 1984). Principles of S-R compatibility with spatial and verbal tasks: The role of display-control location and voice-interactive display-control interfacing. *Human Factors, 26*(5), 533-543.

Williams, J. P. (1962). A test of the all-or-none hypothesis for verbal learning. *Journal of Experimental Psychology, 64*, 158-165.

Ziegler, J. E., Hoppe, H. U., & Fahnrich, K. P. (1986). Learning and transfer for text and graphics with a direct manipulation interface. In M. Mantei and P. Orbeton (Eds.), *CHI'86 Human Factors in Computing Systems (Boston, April 13-17)*. New York: ACM, 72-77.

Office of Naval Research
Perceptual Science programs - Code 1142PS
Technical Reports Distribution List

OSD

Dr. Earl Alluisi
Office of the Deputy Under Secretary of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

DEPARTMENT OF THE NAVY

Aerospace Psychology Department
Naval Aerospace Medical Research lab
Pensacola, FL 32508

Aircrew Systems Branch
Systems Engineering Test Directorate
U. S. Naval Test Center
Patuxent River, MD 20670

Mr. Philip Andrews
Naval Sea Systems Command
NAVSEA 61R2
Washington, D.C. 20362

LCDR R. Carter
Office of Chief on Naval Operations
OP-(##D#
Washington, D.C. 20350

Dr. L. Chmura
Computer Sciences & Systems
Code 5592
Naval Research Laboratory
Washington, D.C. 20350

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 North Quincy Street
Arlington, VA 22217-5000

Commander
Naval Air Systems Command
Crew Station Design
NAVAIR 5313
Washington, D.C. 20361

Dean of Academic Departments
U. S. Naval Academy
Annapolis, MD 21402

Director
Technical Information Division
Code 2627
Naval Research Laboratory
Washington, D.C. 20375-5000

Dr. Robert A. Fleming
Human Factors Support Group
Naval Personnel Research &
Development Center
1411 South Fern Street
Arlington, VA 22202-2896

Mr. Jeff Grossman
Human Factors Laboratory, Code 71
Navy Personnel R&D Center
San Diego, CA 92152-6800

Human Factors Branch
Code 3152
Naval Weapons Center
China lake, CA 93555

Human Factors Department
Code N-71
Naval Training Systems Center
Orlando, FL 32813

Human Factors Engineering
Code 441
Naval Ocean Systems Center
San Diego, CA 92152

CAPT Thomas Jones
Code 125
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

Mr. Todd Jones
Naval Air Systems Command
Code APC-2050
Washington, D.C. 20361-1205

Dr. Michael Letsky
Office of the Chief of Naval
Operations (OP-01B7)
Washington, D.C. 20350

Lt Dennis McBride
Human Factors Branch
Pacific Missle Test Center
Point Mugu, CA 93042

LCDR Thomas Mitchell
Code 55
Naval Postgraduate School
Monterey, CA 93940

Dr. George Moeller
Human Factors Department
Naval Submarine Medical
Research lab
Naval Submarine Base
Groton, CT 06340-5900

CAPT W. Moroney
Naval Air Development Center
Code 602
Warminster, PA 18974

Dr. A. F. Norcio
Computer Sciences & Systems
Code 5592
Naval Research Laboratory
Washington, D.C. 20375-5000

CDR James Offutt
Office of the Secretary of Defense
Strategic Defense Initiative Organization
Washington, D.C. 20301-5000

Perceptual Sciences Program
Office of Naval Research
Code 1142PS
800 North Quincy Street
Arlington, VA 22217-5000

LCDR T. Singer
Human Factors Engineering Division
Naval Air Development Center
Warminster, PA 18974

Dr. A. L. Slafkosky
Scientific Advisor
Cammandant of the Marine Corps
Washington, D.C. 20380

Special Assistant for Marine
Corps matters
Code 00MC
Office of Naval Research
Code 1142PS
800 North Quincy Street
Arlington, VA 22217-5000

DEPARTMENT OF THE ARMY

Director, Organizations and Systems
Research Laboratory
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Michael Drillings
Basic Research Office
U.S. Army Research Institue
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Edgar M. Johnson
Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Technical Director
U.S. Army Human Engineering Laboratory
Aberdeen Proving Ground, MD 21005

DEPARTMENT OF THE AIR FORCE

Mr. Charles Bates, Director
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

Dr. Kenneth R. Boff
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

OTHER GOVERNMENT AGENCIES

Defense Technical Information
Center
Cameron Station, Bldg. 5
Alexandria, VA 22314

Dr. Clinton Kelly
Defense Advanced Research
Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Alan Leshner
Division of Behavioral and
Neural Sciences
National Science Foundation
1800 G. Street, N.W.
Washington, D.C. 20550

Dr. M. C. Montemerlo
Information Sciences &
Human Factors Code RC
NASA HQS
Washington, D.C. 20546

OTHER ORGANIZATIONS

Dr. H. E. Bamford
Program Director
Division of Information, Robotics and
Intelligent Systems
National Science Foundation
Washington, D.C. 20550

Dr. Jay Elkerton
University of Michigan
Dept of Industrial & Operations Engr.
Center for Ergonimics
1205 Beal Avenue
Ann Arbor, MI 48109

Dr. James H. Howard, Jr.
Department of Psychology
Catholic University
Washington, D.C. 20064

Dr. Thomas G. Moher
Dept. of Electrical Engineering
and Computer Science
University of Illinois at Chicago
P.O. Box 4348
Chicago, IL 60680

Dr. Allen Newell
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Jesse Orlansky
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

Dr. Scott Robertson
Department of Psychology
Rutgers University
Busch Campus
New Brunswick, NJ 08903

Dr. H. P. Van Cott
NAS-National Research council
(COHF)
2101 Constitution Avenue, N.W.
Washington, D.C. 20418

END

DATE

FILMED

9-88

DTIC